LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-JRNL-232594

Adaptive Sampling in Hierarchical Simulation

J. Knap, N. R. Barton, R. D. Hornung, A. Arsenlis, R. Becker, D. R. Jefferson

July 9, 2007

International Journal for Numerical Methods in Engineering

Adaptive Sampling in Hierarchical Simulation

J. Knap, N. R. Barton, R. D. Hornung,
A. Arsenlis, R. Becker, and
D. R. Jefferson

Lawrence Livermore National Laboratory
Livermore, CA 94550, USA

July 8, 2007

Abstract

We propose an adaptive sampling methodology for hierarchical multi-scale simulation. The method utilizes a moving kriging interpolation to significantly reduce the number of evaluations of finer-scale response functions to provide essential constitutive information to a coarser-scale simulation model. The underlying interpolation scheme is unstructured and adaptive to handle the transient nature of a simulation. To handle the dynamic construction and searching of a potentially large set of finer-scale response data, we employ a dynamic metric tree database. We study the performance of our adaptive sampling methodology for a two-level multi-scale model involving a coarse-scale finite element simulation and a finer-scale crystal plasticity based constitutive law.

1 Introduction

In recent years, the field of multi-scale materials modeling (MMM) has attracted considerable attention from both the materials science and engineering communities [5, 19, 27, 32, 33, 35]. The adoption of the MMM paradigm has been largely fueled by a wide-spread desire to replace material models entangled in empiricism and coarse phenomenology by an entirely new breed of models. This new class of material models derives its strength from a thorough analysis of the fundamental mechanisms underlying the inelastic behavior of materials. Such analysis naturally involves the entire range of material behaviors. Commonly, this range is conveniently divided into a hierarchy of length scales, and the analysis proceeds by identifying specific unit processes operating at each length scale. A unit process refers to a physical process active at a given length scale that is irreducible and operates independently from all other length scales in the hierarchy [32]. In a multi-scale material model all of the unit processes operate in concert: a unit process at one scale defines boundary conditions and driving forces for the immediately finer scale unit process, but also represents a filtered response of all finer scale processes. For a small class of materials the process hierarchy is relatively well defined. In such cases, the construction of the full multi-scale model is fairly straightforward: the analysis of each unit process may be carried out in turn, followed by the computation of appropriate averages. In general, the process of identification and characterization of unit processes requires the use of diverse analytical or numerical tools, often making the effort prohibitively expensive, or plainly impracticable.

An alternative MMM approach has been proposed by Theodoropoulos et al. [43] (c.f. [23] for an overview of recent developments). This, so-called equation-free (EF), approach, attempts to side-step entirely the difficulties inherent in the identification of unit processes by recourse to direct computation. The properties of EF have been usually investigated in the context of two-scale material models, where the macroscopic, or "coarse", physical process relies on a microscopic, or "fine", process. The fine model is employed to extract relevant microscopic averages required by the macroscopic model. These averages may include additional information in the form of rates of certain quantities, gradients with respect to parameters, Hessians, and so forth. Frequently, computation of such averages involves performing an actual microscopic simulation constrained to a small region of space-time; i.e., a small spatial domain for a short time relative to the coarse-scale dynamics. After averages of the fine-scale behavior are computed, they are used to inform the macroscopic model, which further evolves in time.

Regardless of the manner in which microscopic averages are calculated, the underlying assumption of any MMM is the existence of a response function at each critical length scale in the hierarchy. This function takes coarse-scale driving forces as arguments and returns suitable microscopic averages. In many engineering or material science applications, a certain level of smoothness in the response function may safely be assumed. In such cases, an interpolation procedure may be employed to circumvent the need for an explicit evaluation of the response function in all instances where this information is needed by the coarse-scale model. In 1997, Pope [36] introduced *in-situ* adaptive tabulation (ISAT), a novel adaptive interpolation technique, and applied it to problems in turbulent combustion. Recently, Arsenlis et al. [2] have employed ISAT in the context of constitutive modeling of solids in finite element applications. While only a fraction of constitutive laws may be derived on the basis of a multi-scale modeling paradigm, virtually all constitutive laws may be cast in the form of a generalized response function. Thus, for example, conventional finite element methods may be regarded as a two-scale model by virtue of treating a material constitutive law as the fine-scale model. Moreover, constitutive laws for complex materials are notoriously expensive to evaluate and may dominate the cost of an entire finite element simulation [2]. Therefore, substantial computational savings may be expected from the application of approximation techniques for a broad set of constitutive laws.

In this article, we develop an alternative approach to ISAT methodology employed by Arsenlis et al. [2] in their study of a two-scale (fine-coarse) hierarchical model. We demonstrate that a multi-variate kriging interpolation, in conjunction with a metric-tree database, offers a significant advantage over ISAT in terms of the reduction in the number of required fine-scale model evaluations. Furthermore, we provide a detailed analysis of the performance of the new method in terms of the number of fine-scale model evaluations and error control. While the method is applied here to constitutive model approximation, it is applicable to the more general class of multi-scale embedding techniques.

2 Hierarchical multi-scale model

In the multi-scale hierarchy of models, the interaction between models at two directly neighboring length scales gains a particular significance. This significance is induced, and to a large extent enforced, by the MMM paradigm itself. Regardless of the depth and degree of branching in an overall multi-scale hierarchy, each response function relationship joins two scales with finer-scale information being used to help inform the coarser-scale behavior. Consequently, we restrict our attention in subsequent developments exclusively to this two-scale building block. We wish to emphasize, however, that in doing so we do not impose any constraints on the overall multi-scale model, as the model response at any length scale can be obtained simply by composing model responses of all sub-scale models.

The two-scale model encapsulates a mutual interaction between the coarse-scale model and the fine-scale model (Figure 1). The essence of this interaction is the computation of the response function. Following Arsenlis et al. [2], we view the response function as the mapping

$$\mathbf{o} = \mathcal{M}[\mathbf{i}]\,. \tag{1}$$

Here, $\mathbf{i} \in I$ and $\mathbf{o} \in O$ denote elements in the domain and range of the response function, respectively. Both I and O are assumed to be sets in corresponding vector spaces. Henceforth, without loss of generality, we will identify these vector spaces with $\mathbb{R}^n$ and $\mathbb{R}^m$, where n and m are the dimensions of the domain and range space, respectively. We refer to the elements of set I as points, and of set O as values. Figure 1, represents the essential information exchange in a two-model hierarchy. Note that, in some cases, the fine-scale model may compute $\mathbf{J} = \frac{\partial \mathbf{o}}{\partial \mathbf{i}}$.

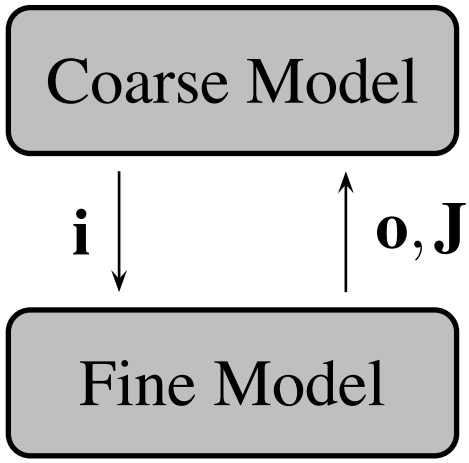


Figure 1: Schematic representation of the two-scale hierarchical model.

The response function (1) is commonly only subject to constraints imposed by the physical nature of the particular problem under consideration. For instance, the response function encapsulating a constitutive law for a material is required to satisfy the principle of material-frame indifference. Moreover, the response function may have arbitrary complexity, and, in many instances, may not be available explicitly. We therefore adopt the view that a detailed knowledge of the internal workings of the response function is not necessary, or even desirable.

In general, the value furnished by the response function (1) may depend on the past history through internal state variables associated with a coarse-scale model or material location. The use of internal variables is common in the mechanics of materials, in particular, to allow for explicit tracking of the micro-structure evolution [30, 34]. Since these state variables are associated with coarse-scale locations, it is inappropriate for the fine-scale model to store them and they are included in each response input vector $\mathbf{i}$. Updated state variable quantities are then included as necessary in the associated response value $\mathbf{o}$. We further allow the response output vector $\mathbf{o}$ to contain additional information, such as quantities used to drive state variable evolution algorithms in the coarse-scale model.

3 Hierarchical two-scale model with adaptive sampling

In the course of a multi-scale simulation involving a two-scale model hierarchy, the coarse model repeatedly requests the evaluation of the response function (1) as it explores the domain I. An example of this behavioral pattern occurs during the solution procedure of an initial boundary value problem (IBVP) in solid mechanics by means of explicit finite elements. In this case, the finite element time integration procedure may be identified with the coarse model and the fine-scale model with a representation of the material constitutive model. The constitutive model is evaluated at each integration point within a finite element. More importantly, this procedure is performed many times during the time evolution of the IBVP. Thus, when the computational cost associated with the evaluation of the constitutive model is large, a substantial fraction of the overall simulation expense may be due to the constitutive model evaluation. For sufficiently expensive constitutive models, the total simulation cost can become prohibitive. However, many problems involve a localized process zone, such as a shock-wave, moving through a solid body. In such cases, the portion of the input space I actively sampled by the coarse model is relatively small [2, 4, 7, 36]. In other cases, the input space is traversed slowly in some combination of the coarse-scale physical space and time dimensions. In either case, it is likely that the cost of the response function computation may be reduced by employing an adaptive interpolation strategy in which response values are approximated in terms of previously-calculated response data. An attractive feature of such an adaptive interpolation strategy is that the number of response function computations does not increase rapidly with increasing resolution of the coarse-scale discretization. By a slight abuse of the nomenclature, we refer to such a strategy as *adaptive sampling* [44].

In an implementation of the adaptive sampling approach, we envision the adaptive sampling procedure to be orchestrated by a single module that interacts with both the coarse and fine models (Figure 2). Like a traditionally-implemented response function, this module accepts requests for response information from the coarse model and supplies the required values. However, the sampling module also dynamically determines whether to interpolate values from previously-computed response data or to invoke the fine model to compute responses directly. Ideally, the adaptive sampling and interpolation process should be transparent to both the coarse and fine models and should not require modifications to either. In practice, some modifications may be required, for example, to allow for slight deviations in the coarse model from a single valued interpolated response as the interpolants are refined. In this context, the set of points at which the value of the response function may be sought during the entire simulation is not predetermined. The extent of this set of points typically increases over the course of a simulation, as the coarse model requires information in an increasingly-larger domain of the fine-scale model. However, the points may remain confined within a relatively small subset or manifold within the total input domain [41]. Since computing responses over the entire input domain can be intractable, we will reject a family of sampling techniques that rely on the construction of a set of point-value response pairs prior to the simulation [36, 42]. We foresee the following scenario for the adaptive sampling module. The operation of the adaptive sampling module commences upon a request from the coarse-scale model for a value $\mathbf{o}$ of the response function at a given point $\mathbf{i}$. Using previously computed values, the module determines whether interpolation at $\mathbf{i}$ will produce a sufficiently accurate approximation of $\mathbf{o} = \mathcal{M}[\mathbf{i}]$. Successfully interpolated results are returned to the coarse model without the need for fine-scale model evaluation. Otherwise, the response function is computed explicitly, and the pair $(\mathbf{i}, \mathbf{o})$ is stored in the adaptive sampling module to inform future calculations.

The efficacy of adaptive sampling hinges on an accurate and fast interpolation scheme. A chief requirement for such an interpolation scheme is the ability to continuously adapt to the evolving set of desired values. Thus, an interpolation scheme using global support, that is one involving all previously computed values at each input point, is rarely appropriate due to the potentially large input set and high cost of re-

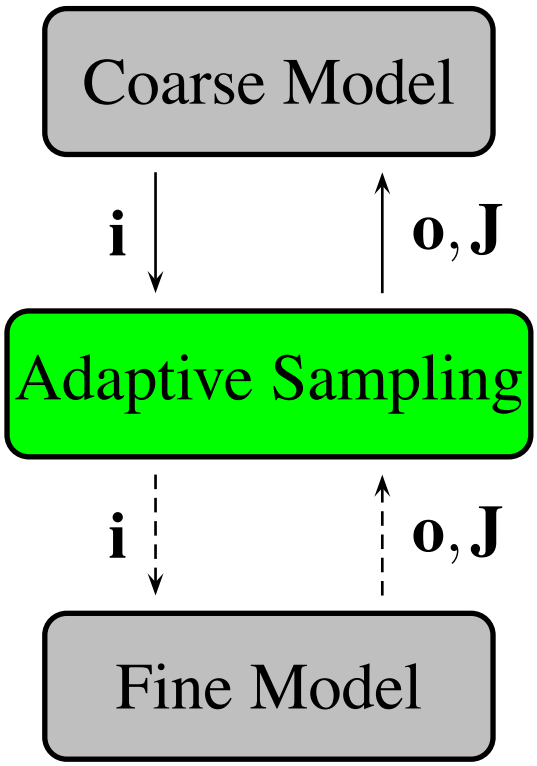


Figure 2: Schematic of the two-scale hierarchical model with adaptive sampling. The adaptive sampling module is drawn in green.

peated construction. Instead, we opt for a local interpolation scheme in which the set of all computed values is divided into a finite collection of subsets, each of which involves a relatively small number of closely spaced values. By virtue of reduction in the number of values involved, local interpolation lowers substantially the cost of building and updating response models. However, this reduction in cost is offset by the need to search a potentially-large set of local interpolants for the most suitable one. Commonly, each of the local interpolants may be assigned a center point, and the search may simply require determination of the interpolant with center closest to the query point. In this context, interpolation is therefore a two-step process: first, a suitable local interpolation model is located by virtue of spatial proximity, and second, the local interpolation scheme is applied to obtain the value. These two tasks constitute the core functionality of response approximation within our adaptive sampling strategy, and their detailed description is provided in the subsequent sections.

4 Response Approximation

The first key component of the adaptive sampling module is the use of multivariate interpolation to avoid repeated evaluation of the response function (1). Here, we provide only a brief summary of major issues (c.f. [1, 9, 17, 18] for an in-depth review). We consider the following data interpolation problem: given

$$(\boldsymbol{x}^i, \boldsymbol{y}^i) \in \Omega \times \Theta,\ i = 1, 2, \ldots, N,\ \Omega \subset \mathbb{R}^n,\ \Theta \subset \mathbb{R}^m, \tag{2}$$

find $\boldsymbol{s} : \Omega \to \Theta,\ \boldsymbol{s} \in S(\Omega)$ such that

$$\boldsymbol{s}(\boldsymbol{x}^i) = \boldsymbol{y}^i,\ i = 1, 2, \ldots, N. \tag{3}$$

We refer to a pair $(\boldsymbol{x}^i, \boldsymbol{y}^i)$ of points $\boldsymbol{x}^i = [x_1^i, \ldots, x_n^i]$ and function values $\boldsymbol{y}^i = [y_1^i, \ldots, y_m^i]$ as the data, where n and m are the dimensionality of the point and function value spaces, respectively. We denote by N the number of data to be interpolated. Ω and Θ represent suitable sets in the corresponding spaces. The interpolating space, $S(\Omega)$, is a finite dimensional vector space of functions defined on Ω. An interpolation scheme is fully specified by prescribing an interpolation space S and an interpolant $\boldsymbol{s} \in S$. Condition (3) expresses the Kronecker-delta property of the interpolant $\boldsymbol{s}$.

4.1 Multivariate interpolation

At present, no multivariate interpolation technique can be considered general and applicable to a wide range of problems. Surprisingly, even in the case of bivariate interpolation, the number of available interpolation methods is truly astonishing [9, 17, 18]. In view of the above, the choice of a multivariate interpolation technique should be decided on the basis of the following considerations:

1. *Locality of the scheme*; In a local scheme, the value of the interpolant at a given location depends solely on a small subset of data sites in the vicinity the evaluation point. Commonly, local schemes are considered more attractive due to their superior performance, whereas global schemes tend to be easier to construct.

2. *Accuracy of the scheme*; For example, in the context of polynomial interpolation, the accuracy is customarily associated with the largest number m such that $\boldsymbol{s} = \boldsymbol{p}$ whenever $\boldsymbol{p}$ is polynomial of degree up to m and $\boldsymbol{y}^i = \boldsymbol{p}(\boldsymbol{x}^i),\ i = 1, 2, \ldots, N$.

3. *Smoothness of the interpolant*; Smoothness requirements vary by application; however, first or sometimes higher-order differentiability is desired. We emphasize that the choice of locality of the interpolation scheme may affect its smoothness as not all local schemes are globally continuous, for example.

4. *Data required for the construction of the scheme*; In many applications the only available data are the function values. However, other quantities, such as values of function derivatives at data sites or function and derivative values at points other than data sites, may be necessary in order to satisfy locality or accuracy constraints.

5. *Computational cost*; The computational cost of a interpolation scheme arises from two sources: the construction of the interpolant and the evaluation of the interpolation at a point of interest.

Multivariate interpolation schemes are usually classified in the following categories [1]:

1. *Tensor product schemes*; These methods aim at extending directly the univariate theory. In order to achieve this goal, the interpolation schemes are obtained by tensor products of univariate approximations. The Lagrange and the Newton formulas with divided differences may be extended easily in this way. However, by virtue of their structure, schemes based on the tensor products require a very special distribution of data points (regular grids, natural lattices, or the like).

2. *Point schemes*; These methods assume that the interpolation spaces are constructed solely on the basis of arbitrarily distributed set of data points, and do not require a tessellation of underlying domain Ω. The most notable schemes in this class include: the family of Shepard's method, radial interpolants, and Duchon thin plate splines.

3. *Natural neighbor interpolation*; This class of methods relies on the Voronoi tessellation of the underlying point set to construct an interpolation scheme.

4. *Simplicial schemes*; These methods rely on the n-dimensional simplicial discretizations of the domain Ω. The simplicial schemes may be formulated as polynomial or rational.

5. *Multivariate splines*; These methods derive a multivariate interpolant by generalizing the concept of univariate splines. As in the univariate case, they require a tessellation of the domain Ω.

As noted above, the fine model may provide both $\mathbf{o}$ and derivative information in some cases. The process of building an interpolant employing the derivative data, along with the functional data, is frequently referred to as *derivative estimation*. In general, the derivative estimation will aid in obtaining an interpolant that is both local and smooth. However, most existing derivative estimation schemes have been rigorously developed for bivariate problems. Moreover, most multivariate derivative schemes have been created *ad hoc*, from the viewpoint of convenience rather than desirable mathematical properties. Therefore, these schemes do not generally perform well on arbitrary problems [1]. The most suitable member of a collection of such interpolation methods is usually determined by means of numerical experiments.

The development of multivariate interpolation schemes is a rich and rapidly evolving research area. While highly sophisticated, complex, and powerful schemes exist in a variety of forms, they tend to carry high computational cost (e.g. natural neighbor interpolation, simplicial schemes) or provide acceptable accuracy only for a moderate number of data (e.g. higher dimensional polynomial interpolants). In addition, the multivariate interpolation schemes applied in practice are traditionally *ad hoc* schemes such as Shepard's method [40].

4.2 Kriging interpolation

Over the years, a viable alternative to the conventional multivariate interpolation techniques has emerged in the field of geostatistics [22, 47]. This technique, commonly referred to as kriging, has been successfully applied to practical problems in a wide array fields outside geostatistics, such as analysis of computer experiments [31, 38], design optimization [10, 48], and numerical methods [20]. Kriging is a multivariate smooth global point interpolation scheme possessing a very desirable feature in practical applications: apart from providing interpolated value, it also yields a mean squared estimate of the interpolation error.

4.2.1 Univariate interpolation

For simplicity of introduction to the salient points of kriging interpolation, we describe univariate interpolation, which corresponds to $n = m = 1$ in (2). Subsequently, an extension to the multivariate kriging interpolation is provided. Kriging strives to replace the interpolant $s : \Omega \to \Theta \subset \mathbb{R}$ in the interpolation problem (2) with a combination of a regression model and a deviation:

$$s(\boldsymbol{x}) = f(\boldsymbol{x}) + Z(\boldsymbol{x}),\ \boldsymbol{x} \in \Omega. \tag{4}$$

Here, $f : \Omega \to \mathbb{R}$ denotes the regression model and $Z : \Omega \to \mathbb{R}$ is a realization of a random function (stochastic process). Ordinarily, the regression model alone lacks the Kronecker-delta property (3). Hence, the role of the random process $Z(\boldsymbol{x})$ is to generate a localized deviation from the regression model in order to induce the Kronecker-delta property over the interpolant s.

Commonly, the regression model is assumed to posses the following linear structure:

$$f(\boldsymbol{x}) = \sum_{i=1}^{D} p_i(\boldsymbol{x})\beta_i \equiv \boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{\beta}, \tag{5}$$

where: β_i are unknown regression coefficients and $p_i : \Omega \to \mathbb{R}$ are the regression basis functions. Here, $\boldsymbol{v}^T$ denotes the transpose of a vector $\boldsymbol{v}$. A linear basis in one dimension is given by

$$\boldsymbol{p}(x) = [1, x]^T, \tag{6}$$

and in two dimensions assumes the following form

$$\boldsymbol{p}(\boldsymbol{x}) \equiv \boldsymbol{p}(x_1, x_2) = [1, x_1, x_2]^T. \tag{7}$$

The random process, $Z(\boldsymbol{x})$, is assumed to have zero mean and covariance of the following form:

$$\mathrm{Cov}\left[Z(\boldsymbol{x}), Z(\boldsymbol{w})\right] = \sigma^2 R(\boldsymbol{x}, \boldsymbol{w}). \tag{8}$$

Here, σ^2 is the process variance, $R : \Omega \times \Omega \to \mathbb{R}^+$ is the stochastic correlation function, and $\mathbb{R}^+$ denotes the non-negative real numbers. Popular choices for function R include: exponential, Gaussian exponential, or spline (see [22, 47] for a general discussion and properties of various correlation models).

The best linear unbiased prediction of the Eq. (4) at $\boldsymbol{x} \in \Omega$ is obtained by minimizing the mean squared error over the random process [38], leading to the following expression

$$\hat{s}(\boldsymbol{x}) = \boldsymbol{p}^T(\boldsymbol{x})\hat{\boldsymbol{\beta}} + \boldsymbol{r}^T(\boldsymbol{x})\boldsymbol{R}^{-1}(\boldsymbol{v} - \boldsymbol{P}\hat{\boldsymbol{\beta}}). \tag{9}$$

Here, $\hat{\boldsymbol{\beta}}$ is given by

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{P}^T\boldsymbol{R}^{-1}\boldsymbol{P})^{-1}\boldsymbol{P}^T\boldsymbol{R}^{-1}\boldsymbol{v}, \tag{10}$$

where

$$\boldsymbol{P} = \begin{bmatrix} p_1(\boldsymbol{x}^1) & \cdots & p_D(\boldsymbol{x}^1) \\ \cdots & \ddots & \cdots \\ p_1(\boldsymbol{x}^N) & \cdots & p_D(\boldsymbol{x}^N) \end{bmatrix}, \tag{11}$$

$\boldsymbol{v} = [y^1, \cdots, y^N]$ is the vector of data, and $\boldsymbol{A}^{-1}$ indicates the inverse of matrix $\boldsymbol{A}$. $\boldsymbol{R}$ in equation (9) is the correlation matrix defined as

$$\boldsymbol{R} = \begin{bmatrix} 1 & R(\boldsymbol{x}^1, \boldsymbol{x}^2) & \cdots & R(\boldsymbol{x}^1, \boldsymbol{x}^N) \\ R(\boldsymbol{x}^2, \boldsymbol{x}^1) & 1 & \cdots & R(\boldsymbol{x}^2, \boldsymbol{x}^N) \\ \cdots & \cdots & \ddots & \cdots \\ R(\boldsymbol{x}^N, \boldsymbol{x}^1) & \cdots & R(\boldsymbol{x}^N, \boldsymbol{x}^{N-1}) & 1 \end{bmatrix}, \tag{12}$$

and

$$\boldsymbol{r}(\boldsymbol{x}) = \begin{bmatrix} R(\boldsymbol{x}^1, \boldsymbol{x}) \\ \cdots \\ R(\boldsymbol{x}^N, \boldsymbol{x}) \end{bmatrix}. \tag{13}$$

One of the most appealing features of the kriging interpolation is its ability to provide, in addition to the value of the interpolant, an estimate of the interpolation error. This estimate takes on the form of the mean squared error given by the following expression:

$$e(\boldsymbol{x}) = \sigma^2 \left[1 + \boldsymbol{u}(\boldsymbol{x})^T (\boldsymbol{P}^T \boldsymbol{R}^{-1} \boldsymbol{P})^{-1} \boldsymbol{u}(\boldsymbol{x}) - \boldsymbol{r}^T(\boldsymbol{x}) \boldsymbol{R}^{-1} \boldsymbol{r}(\boldsymbol{x})\right], \tag{14}$$

where $\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{P}^T \boldsymbol{R}^{-1} \boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{p}(\boldsymbol{x})$, and the process variance, σ^2, is calculated according to

$$\sigma^2 = \frac{(\boldsymbol{v} - \boldsymbol{P}\hat{\boldsymbol{\beta}})^T \boldsymbol{R}^{-1} (\boldsymbol{v} - \boldsymbol{P}\hat{\boldsymbol{\beta}})}{N}. \tag{15}$$

Construction of the kriging interpolant (9), together with the mean squared error estimate (14), requires specifying the regression model and the stochastic correlation function. After these two data are specified, the construction of (9) is fairly straightforward. In addition, factors in (9) and (14) independent of $\boldsymbol{x}$, such as $\boldsymbol{R}^{-1}$, $(\boldsymbol{P}^T \boldsymbol{R}^{-1} \boldsymbol{P})^{-1}$ or $\hat{\boldsymbol{\beta}}$, can be stored during the construction of the interpolant for subsequent reuse when the value of the interpolant and associated error estimate are calculated. An in-depth analysis of the associated cost may be found in [28, 29].

4.2.2 Multivariate interpolation

In the context of kriging, multivariate interpolation involves the joint prediction of multiple correlated variables. Ver Hoef and Cressie [46] have shown that the univariate definition of both the kriging interpolant (9) and the mean squared error (14) can be, without much difficulty, extended into the multivariate case. This extension involves minor modifications of $\boldsymbol{p}(\boldsymbol{x})$, $\boldsymbol{P}$, $\boldsymbol{R}$, and $\boldsymbol{r}(\boldsymbol{x})$, details are available in [14, 46, 47].

4.2.3 Derivative estimation

The construction of the kriging interpolant (9) does not account for additional data that may be associated with each data point. Additional quantities, such as derivative information, are usually correlated with the primary variable and thus contain important information about the primary variable itself. The incorporation

of additional information aimed at improving the accuracy of the kriging interpolant is commonly referred to as cokriging.

The cokriging method can be conveniently cast within the framework of multivariate kriging. In the context of the derivative estimation, this procedure relies on the extension of the data vector $\boldsymbol{v}$ to include the values of the derivatives at data points. The sole remaining issue is then the expression for the correlation matrix $\boldsymbol{R}$. Morris et al. [31] start with the covariance matrix of $Z(\boldsymbol{x})$ for the original kriging method defined as

$$\text{Cov}\left[Z(\boldsymbol{x}), Z(\boldsymbol{w})\right] = \sigma^2 R(\boldsymbol{x}, \boldsymbol{w}), \tag{16}$$

where $R(\boldsymbol{x}, \boldsymbol{w})$ is the stochastic correlation function. The cokriging covariance is then obtained by direct differentiation of (16) yielding

$$\begin{aligned}
\text{Cov}\left[Z(\boldsymbol{x}), Z(\boldsymbol{w})\right] &= \sigma^2 R(\boldsymbol{x}, \boldsymbol{w}), & (17)\\
\text{Cov}\left[Z(\boldsymbol{x}), \frac{\partial Z(\boldsymbol{w})}{\partial x_k}\right] &= \sigma^2 \frac{\partial R(\boldsymbol{x}, \boldsymbol{w})}{\partial x_k}, & (18)\\
\text{Cov}\left[\frac{\partial Z(\boldsymbol{x})}{\partial x_k}, Z(\boldsymbol{w})\right] &= -\sigma^2 \frac{\partial R(\boldsymbol{x}, \boldsymbol{w})}{\partial x_k}, & (19)\\
\text{Cov}\left[\frac{\partial Z(\boldsymbol{x})}{\partial x_k}, \frac{\partial Z(\boldsymbol{w})}{\partial x_l}\right] &= -\sigma^2 \frac{\partial^2 R(\boldsymbol{x}, \boldsymbol{w})}{\partial x_k \partial x_l}. & (20)
\end{aligned}$$

By recourse to the above expressions, the cokriging correlation matrix becomes readily available. Subsequently, the multivariate kriging formulation can be directly employed incorporating the newly obtained cokriging correlation matrix.

4.3 Local kriging interpolation

For a set of N data points, and associated values, the procedure of constructing the univariate kriging interpolant (9) requires numerous solutions of linear systems of equations. The largest of these linear systems involves an $N \times N$ covariance matrix $\boldsymbol{R}$. Ordinarily, $\boldsymbol{R}$ is symmetric, but otherwise, it does not possess any special or sparse structure. Hence, with the exception of small data sets, the computational cost of the global kriging interpolant may become substantial even for relatively small values of N. Since adaptive sampling is based upon repeated construction of the interpolant as the new data become available, the expense of repeated matrix inversions severely limits the practicality of the global kriging interpolant in this context. We circumvent this fundamental limitation of global kriging by employing a local kriging interpolant.

Local kriging assumes a certain degree of data locality; i.e., values associated with "distant" points are only weakly correlated. Thus, the domain Ω may be subdivided into S sub-domains, $\{\Omega_1, \Omega_2, \ldots, \Omega_S\}$, not necessarily disjoint. We denote by N_i the number of data points contained in sub-domain Ω_i. A local kriging interpolant, $\hat{s}_i$, can be associated with subdomain Ω_i by constructing it solely on the basis of the N_i data points contained in Ω_i. The distribution of the data points among the sub-domains is assumed exclusive, i.e. $N = \sum_i^S N_i$. It is important to emphasize, however, that the above assumption is not essential in our formulation and it is possible for a number of local kriging interpolants to share a point. We will not employ such local kriging interpolants in this work and defer these developments to subsequent work.

To further reduce the cost of creation and storage of kriging interpolation models, we assume that there exists a user-imposed upper bound on the number of data points contained in each of the sub-domains, N_{max}, i.e. $N_i \leq N_{max},\ i = 1, \ldots, S$. However, in general, the sub-domains need not contain identical numbers of data points. Thus, local kriging interpolants containing N_{max} data points may transition into,

essentially, a frozen state. Conversely, those local kriging interpolants containing less that N_{max} data points may still be allowed to change and grow. This feature conforms well to the goal of adaptive sampling as regions of domain Ω that are relatively dense with data points will tend to contain a majority of frozen local kriging models, whereas those regions being actively explored by the coarse model tend to be sparsely populated with data points, and thus will contain fewer such interpolants.

The use of local kriging interpolants allows for a substantial reduction in computational expense with respect to the global interpolant. This reduction, however, is achieved at an additional expense of searching for a suitable local interpolant at each point $\boldsymbol{x}$ where a value is needed. A variety of methods for determining a suitable local interpolant are possible. Here, we adopt what may be the most straightforward choice: for point $\boldsymbol{x}$ the most suitable local interpolant is the one whose "distance", induced by a suitable metric in $\mathbb{R}^n$, from $\boldsymbol{x}$ is the smallest among all local interpolants. The distance between $\boldsymbol{x}$ and local interpolant $\hat{s}_i$ is defined as the distance between $\boldsymbol{x}$ and the centroid of N_i points forming the basis of the local interpolant. In practice, we modify the above approach and select the most suitable local kriging interpolant among a small number of nearest local interpolants.

In summary, the process of computing the interpolated value for a point $\boldsymbol{x}$ via local kriging is composed of two stages. In the first stage, the sub-domain Ω_i closest to $\boldsymbol{x}$ is located. Afterward, the local kriging model associated with Ω_i is employed for interpolation. Therefore, the efficiency of the local kriging approach depends on the performance of the local kriging interpolation, on the ability to store continuously evolving local kriging models, and on the speedy determination of the closest local kriging model. We address these issues in the subsequent section.

5 Response Data Management

The local kriging interpolation is computationally less expensive than a global kriging interpolation scheme. However, a primary concern for utilizing local approximation models in adaptive sampling is that the cost associated with storing and retrieving local models must not offset their savings nor the savings achieved through the elimination of a significant fraction of fine-scale model evaluations. Thus, a central ingredient in the adaptive sampling methodology is a database mechanism for managing a potentially large, dynamically generated collection of local models.

5.1 Goals and Issues

A practical database scheme to be employed in the current context must yield efficient storage, retrieval, and modification of the set of local models so that these data management operations are small compared to the cost of local kriging model construction. In particular, we would like such a database to possess the following properties:

- The database must be dynamic and support fast and frequent insertion, deletion, and modification of data elements. In particular, incremental modification of the data set must not require substantial changes to the structure used to organize it.
- The database should not require *a priori* knowledge of the extent of a data set nor should it require an *a priori* data partitioning to be defined.
- The database should scale well as the number of stored local kriging interpolants increases.

- Data must be stored in and accessible from both main memory and disk. To reduce data storage for large simulations, frequently accessed local kriging interpolants should reside in memory while those that are accessed occasionally can reside on disk.
- Database performance should only weakly depend on the dimensionality of both the point and value spaces so that the database strategy may be employed across a wide range of problems.
- It should be inexpensive to maintain *balance* in the structure so that most queries and data insertions have comparable I/O and computational costs.
- Nearest-neighbor (NN) and range queries are most important for the applications of interest to us. Since query performance in high-dimensional data sets typically depends on the cost of *distance* computations used to compare data items, the number of distance computations needed to insert and retrieve data items should not be excessive.
- Since the choice of an appropriate distance function often depends on the data set generated by an application, the database structure should allow for alternative distance functions to be used. Experimenting with distance functions will be necessary to achieve high efficiency in any given simulation.
- The database should support parallelization to the extent that multiple queries may execute concurrently since our applications of interest involve parallel simulation codes. Ideally, we would like the database scheme to support distributed memory parallelism so that updates in the data produced by large-scale distributed memory applications do not constitute a serialization bottleneck.

Over the past couple of decades, many database methods have been developed for applications involving multidimensional point data. The surveys [8, 16] describe the wide range and applicability of available methods. Many methods appear to perform well for problems involving specific data distributions and queries on that data. However, it also appears that methods can be sensitive to data distributions for which they are not designed. Data sets that are highly skewed, correlated, or clustered, appear to be especially problematic as the dimensionality of the data increases. There is very little quantitative analysis of these phenomena and most theoretical and empirical analysis assumes that data is uniformly distributed within the data space.

Despite the tremendous progress in the development of novel database algorithms, there is little general guidance available for choosing a data management scheme that is best-suited to a given application. We dismissed many approaches that did not meet certain of our requirements listed above. For example, methods that require an *a priori* partitioning of the data set or are designed primarily for static data sets were eliminated. We also dismissed schemes whose performance appeared to depend heavily on the dimensionality or distribution of a data set. After an extensive exploration of potential candidate spatial index structures, we settled on the "M-tree" method.

5.2 M-tree Database

For managing local model approximations, we adopted the "M-tree", approach introduced in [49] and customized certain aspects of it to suit our needs. An M-tree possesses many of the features we identified earlier for a collection of local kriging interpolants. An M-tree defines a spatial indexing scheme suitable for high-dimensional spatial data that may be distributed arbitrarily within the data domain. Also, an M-tree supports dynamic insertion and deletion of data items. It does not require data preprocessing prior to insertion into

the structure nor does is require specification of a partitioning of the data space *a priori*. Perhaps most importantly, M-trees offer flexibility with regard to various parameters and algorithms that are used to define and manipulate the index structure. This allows us to formulate adaptive sampling algorithmic components that are largely generic with respect to the underlying application. Thus, we may explore variations within that methodology and tune it to the requirements of each application.

5.2.1 M-tree Background

Metric trees were developed to store and search large, high-dimensional data sets using the notion of a general *metric space*. Data object proximity is defined in terms of a distance function; that is, a positive real-valued function that takes two data objects as arguments, is symmetric in those arguments, and satisfies the triangle inequality. Metric trees are designed to reduce the number of distance computations required to insert and search for data objects [45]. Certain information is stored in the tree structure as it is constructed so that relative distances between data objects can be *compared*, to the extent possible, while actual distance *computations* are performed only when necessary.

An M-tree is a particular type of metric tree that supports dynamic insertion of data objects and offers several heuristic options for adapting the structure of the tree to the distribution of data objects as they are added. Studies of M-tree performance on a variety of data sets suggest that they are well-suited to high-dimensional data spaces and scale well operationally as the data sets grow [11, 12, 13]. It is interesting to note that performance studies of spatial index methods tend to focus on I/O costs, specifically the number of data objects accessed during searches, and ignore the expense of object comparisons. However, we have found that the cost of such comparisons, distance computations in particular, can be significant if not managed appropriately. Most M-tree studies appear to consider the number and expense of distance computations as well as I/O costs.

In the next several sections, we describe basic features of an M-tree structure and associated data insertion and searching operations. More detailed information can be found in [11].

5.2.2 M-tree Structure

An M-tree is a hierarchical index structure that contains two types of nodes. Data objects are referenced in *leaf nodes*, which have no children. Non-leaf nodes, called *routing nodes*, store "routing" objects that characterize the structure of their corresponding subtrees. Each data object and routing object is identified by its *feature vector*, which for our purposes, is a point in $\mathbb{R}^n$ where n is the dimensionality of the data points described in Section 4. Recall that the data objects that we store are local kriging interpolants which contain information used to approximate a response function over some region of the response domain. However, the following discussion does not require us to distinguish an object from its feature vector.

We denote data objects and routing objects (i.e., feature vectors) by D and R, respectively. The distance between any two objects U and V is indicated by $\text{dist}(U, V)$. Each routing object R is associated with a subtree $T(R)$, called the *covering tree* of R, and a *covering radius* $\text{rad}(R)$. Each object residing in the subtree $T(R)$ satisfies the *covering property*: $\text{dist}(R, V) + \text{rad}(V) < \text{rad}(R)$. The covering property is an essential aspect of the M-tree as it is central to operations involving tree traversal.

An M-tree organizes the data space into a recursive hierarchy of overlapping regions. Each tree node contains multiple *entries*, each of which is associated with a spherical region centered at an object. The number of entries in a node may be fixed or variable; this value determines the height and fan-out of the tree. To facilitate data insertion and searching operations, each entry holds information, in addition to an

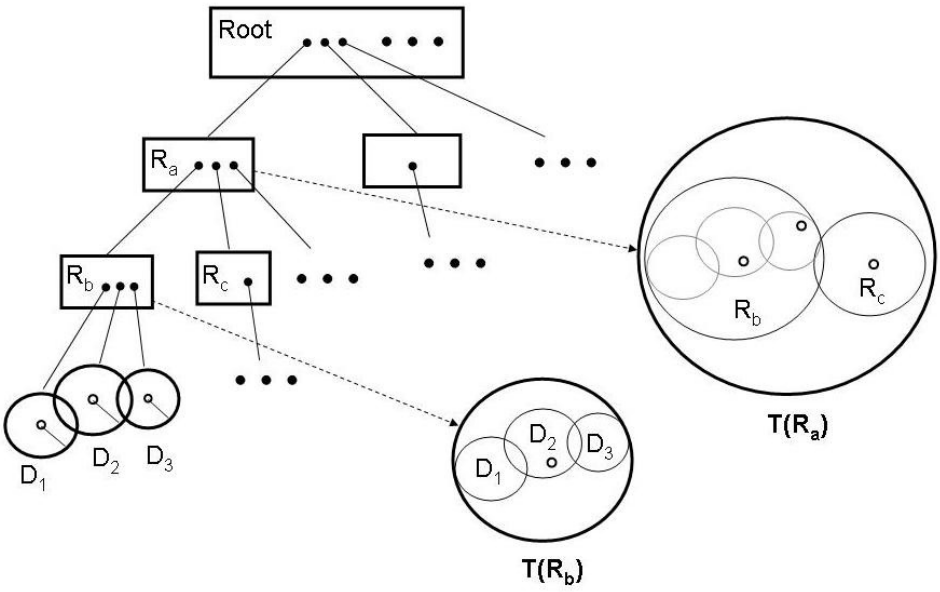


Figure 3: The M-tree defines a hierarchical covering of regions in the underlying data space. Each tree entry defines a region with a certain radius centered at the corresponding object (i.e., feature vector). The region of a non-leaf (i.e., routing) entry contains all regions in its covering tree. Here, R_a is a routing entry in the root node corresponding to a routing node one-level down and containing entries routing R_b and R_c. The covering tree $T(R_a)$ contains the covering trees $T(R_b)$ and $T(R_c)$. R_b contains data entries D_1, D_2, and D_3, the regions of which are contained in $T(R_b)$.

object. Specifically, each routing entry is defined by

$$\text{entry}(R) = \{R, T(R), \text{rad}(R), \text{dist}(R, \text{par}(R))\} \tag{21}$$

and each data entry is defined by

$$\text{entry}(D) = \{D, \text{data}(D), \text{dist}(D, \text{par}(D))\} \ . \tag{22}$$

Here, $\text{par}(V)$ is the object associated with the parent node containing the object V. So a routing entry defines a spherical region in the metric space centered at the corresponding routing object R and having radius $\text{rad}(R)$. Figure 3 illustrates the essential features of the M-tree structure.

All objects corresponding to entries in a node have the same parent object and the entries are sorted in order of increasing distance from their objects to the parent. This sorting and the covering property enable the triangle inequality to be used to avoid distance computations when possible thus increasing the efficiency of operations involving tree traversal. Note that $\text{dist}(R, \text{par}(R))$ is not defined when R is an object in the root node of the tree. Consequently, operations starting at the root node are the only instances in which it is always necessary to compute the distance to every object in a node.

5.2.3 Adding Data Objects to an M-tree

Adding a data object D to an M-tree requires two main procedures: one that determines the most suitable leaf node for the object and one that splits a node when it becomes full. Finding a suitable leaf node is

a recursive process starting at the root node. At each node along the way from the root to the leaf level, one attempts to find a routing entry entry(R) so that descending into its subtree $T(R)$ does not enlarge the associated covering radius; that is, dist(R, D) + rad$(D) \leq$ rad(R). If there are multiple subtrees satisfying this inequality, then $T(R)$ for which dist(R, D_{new}) is minimized is chosen. This choice maintains *well-clustered* subtrees. If no subtree satisfies the inequality, then $T(R)$ for which dist(R, D_{new}) − rad(R) is minimized is chosen. This choice is based on the heuristic that the total volume of the covering trees associated with the current node should be minimized.

Like other *balanced* tree structures, an M-tree grows in a "bottom-up" fashion. When the number of data objects held in a leaf node reaches a specified number, the *node capacity*, the node is split. That is, the node is replaced by two new nodes and a routing object is chosen for each new node. This procedure is referred to as object *promotion*. Then, the entries are divided into two disjoint sets each of which is assigned to one of the two new nodes; this procedure is called object *partitioning*. The node splitting process continues recursively up to the parent node as needed. When the root node is split, a new root node is created and the tree grows in height by one level.

The combination of a promotion method and a partitioning method as constitute a *node split policy* for an M-tree. Numerous split policies have been proposed and explored in the analysis of M-tree performance on various data sets. Regardless of the split policy details, the covering property described earlier must be preserved. This implies that, when a node is split, the covering radii of entries in the node along the path from that node to the root may require modification. An important cost consideration for M-tree data insertion involves distance computations, which are potentially expensive in high dimensions, required during node splitting. Since a primary aim of an M-tree structure is to minimize the number of distance computations that are performed, simple split policy schemes that require few distance computations are usually preferred. However, performance studies have also shown that more sophisticated strategies, which require more distance computations, may yield better I/O performance during searches in higher spatial dimensions, due to better tree "quality". Better tree quality is usually indicated by tight object clustering and a low average covered volume of the subtrees associated with entries in a node.

5.2.4 M-tree Searches

M-tree searching performance strongly depends on how well data insertion operations organize the tree structure with respect to the underlying data distribution. M-tree searching algorithms exploit the triangle inequality and pre-computed stored distances and covering radii similarly to data insertion procedures. During a search, the goal is to traverse the tree by accessing no more nodes and by performing no more distance computations than are necessary to find the object(s). Here, we briefly discuss the *kNN-search* algorithm which retrieves the k "nearest neighbors" in the tree to a given query object Q. This search is the most relevant to the adaptive sampling application discussed in Section 7.

The standard kNN-search algorithm for M-trees uses a branch-and-bound technique similar to the commonly-used RKV-algorithm [37]. The kNN-search algorithm uses a priority queue of pointers to *active subtrees* and an element array of length k that will contain the search result. At a routing node, the search determines its active subtrees which are those that cannot be pruned from the search yet; these are inserted into the priority queue. An upper bound dist$_{max}(T(R))$ on the distance between Q and any object in a subtree $T(R)$ defined as

$$\text{dist}_{max}(T(R)) = \text{dist}(R, Q) + \text{rad}(R) \ , \tag{23}$$

is computed for each subtree. When interpreted in the proper order with respect to elements already placed in the array of k nearest neighbors, the upper bounds can be used to prune entire subtrees from the priority

queue and, thus, the search. At each point in the search algorithm, a pointer to some subtree $T(R)$ and a lower bound $\text{dist}_{min}(T(R))$ on the distance between Q and any object in $T(R)$ are also maintained. The lower bound is defined as

$$\text{dist}_{min}(T(R)) = \max\{\text{dist}(R, Q) - \text{rad}(R), 0\} \tag{24}$$

since no object in $T(R)$ can be closer to Q than $\text{dist}(R, Q) - \text{rad}(R)$. This bound is used to extract the next node to be examined from the priority queue. Since the order in which nodes affects the query performance, the node selected is the one for which $\text{dist}_{min}(T(R))$ is a minimum.

6 Adaptive sampling algorithm

In this section, we provide a detailed description of our adaptive sampling module. The module utilizes local kriging for the interpolation of the response function, as well as the M-tree database for storage and retrieval of appropriate local interpolants. Many details in this section are specific to our demonstration application: a hierarchical two-scale model involving a conventional finite element solver at the coarse scale and an aspect of the constitutive material relation at the fine scale. More specifically, we restrict our focus to explicit finite elements in the context of deformable solids. Thus, we envision the operation of the finite element solver constrained to the following two tasks: i) computing nodal forces from stresses, and ii) employing the nodal forces in a time-stepping algorithm in order to follow the time evolution of nodal positions. The computation of the nodal forces commonly involves a loop extending over the entire discretization of a solid at each time step. During the time step, a constitutive material relation is evaluated at a fixed number of quadrature points within each finite element in the discretization[6, 21]. In addition, we assume that the response function also returns gradient data as a part of its output. This gradient data is used in the non-linear solution process for the overall constitutive response at the coarse scale.

The operation of the adaptive sampling module commences with a request for the value of the response function (1) for a given input vector $\mathbf{i} \in I$. Subsequently, local kriging interpolation is attempted to approximate the value of the response function at $\mathbf{i}$. This interpolation step involves, first, locating the most suitable local kriging interpolant, and second, using this interpolant to compute an error estimate. The approximated value of the response is used if the error is found to be acceptable. Fortunately, for many engineering applications input vectors for subsequent response function evaluations from one time step to the next at a given quadrature point are often only marginally different. We take this spatial locality (in the sense of I) into account by preceding an M-tree database search with an error calculation using the kriging interpolant associated with the quadrature point from the previous response evaluation. If the error estimate from the previously used interpolant is less than the user specified error tolerance, interpolation is performed without recourse to a search of the M-tree.

Obviously, this procedure is not guaranteed to succeed at all times. Thus, in such a case, we search the M-tree database for kriging interpolants based on spatial proximity of a given input vector $\mathbf{i} \in I$. Since the response function may be arbitrarily complex, the closest local kriging interpolant may not yield the highest quality approximation. Therefore, we query the M-tree database for a small number, C_{max}, of nearest neighbor local kriging interpolants. Subsequently, we scan the set of search results in order of increasing distance from $\mathbf{i}$ for a local kriging interpolant satisfying the user error tolerance. If no such interpolant is available in the search result, an explicit evaluation of the response function is performed. The newly computed value of the response function, $\mathbf{o}$, at $\mathbf{i}$ needs to be stored in the M-tree database. A detailed representation of the adaptive sampling algorithm is given in Algorithm 1.

Algorithm 1: AS($\mathbf{i}, E, Q$)

$K \leftarrow$ LAST(E, Q) // Get interpolant, K, used previously with element E and quadrature point Q
if $K \neq$ NULL **then**
  ERR $\leftarrow$ ERROR$(K, \mathbf{i})$ // Compute error estimate at **i** using K
  if ERR $\leq$ TOL **then**
    $\mathbf{o} \leftarrow$ INTERPOLATE$(K, \mathbf{i})$
    return $\mathbf{o}$
  end if
end if
$\mathbf{K} =$ CLOSEST$(\mathbf{i}, C_{max})$ // Get at most C_{max} closest kriging interpolants to **i** from the M-tree database
for all $K \in \mathbf{K}$ **do**
  ERR $\leftarrow$ ERROR$(K, \mathbf{i})$
  if ERR $\leq$ TOL **then**
    $\mathbf{o} \leftarrow$ INTERPOLATE$(K, \mathbf{i})$
    return $\mathbf{o}$
  end if
end for
$\mathbf{o} \leftarrow$ EVALUATE$(\mathcal{M}[\mathbf{i}])$ // Evaluate the response function
ADDPAIR$(K, \mathbf{i}, \mathbf{o})$ // Store $(\mathbf{i}, \mathbf{o})$ into K
return $\mathbf{o}$

Algorithm 2: ADDPAIR$(K, \mathbf{i}, \mathbf{o})$ Add point-value pair into a local interpolant

ADDPOINT$(K, \mathbf{i}, \mathbf{o})$ // Add point-value pair to the local interpolant
$N_P \leftarrow$ DIM(POINTS(K)) // Get number of point-value pairs in K
if $N_P > N_{max}$ **then**
  $(K_1, K_2) \leftarrow$ SUBDIVIDE(K) // Perform subdivision of K if capacity reached
  DELETE(K) // Remove K from the M-tree database
  $C_1 \leftarrow$ CENTROID(K_1) // Compute center of K_1
  STORE(K_1, C_1) // Store K_1 in the M-tree database at C_1
  $C_2 \leftarrow$ CENTROID(K_2) // Compute center of K_2
  STORE(K_2, C_2) // Store K_1 in the M-tree database at C_2
else
  DELETE(K) // Remove old K from the M-tree database
  $C \leftarrow$ CENTROID(K) // Compute center of updated K
  STORE(K, C) // Store K in the M-tree database at C
end if
return

The newly computed value of the response function, $\mathbf{o}$, in association with $\mathbf{i}$ needs to be stored. However, as mentioned earlier, individual $(\mathbf{i}, \mathbf{o})$ pairs are stored only as a part of a local kriging interpolant. So we add the point-value pair to the closest kriging interpolant in the database and adjust the centroid of the local kriging interpolant. Since the location of the local kriging interpolant in I is likely to change, the process involves removal and subsequent reinsertion of the interpolant in the M-tree. This is described in Algorithm 2. It is worth noting that at the start of a typical simulation the M-tree database is empty. Therefore, when the first point-value pair needs to be stored, a local kriging interpolant containing this single pair is constructed. Such construction is possible due to the presence of the gradient data in $\mathbf{J}$.

The process of repeated insertion of new point-value pairs into a single local kriging interpolant will ultimately lead to its size exceeding the maximum allowed capacity, N_{max}. When this occurs, an obvious approach is to start a new interpolant. However, we adopt a slightly different strategy in which we subdivide the interpolant after the new point-value pair has been added. First, we calculate the second moment of all the points involved in the local kriging interpolant and the eigenvector corresponding to the largest eigenvalue of the second moment, $\hat{\mathbf{n}}$. Subsequently, we construct a hyperplane passing through the centroid of the local kriging interpolant with normal along $\hat{\mathbf{n}}$. This hyperplane naturally partitions the points of the local kriging interpolants into two disjoint sets that, in turn, are used to build two new local kriging interpolants. Then, the original interpolant is removed from the M-tree database and the two new models are inserted. The advantage of this strategy, over simply creating a new local kriging interpolant with a single point, is that it tends to limit overlaps between the domains of local kriging interpolants. Moreover, this strategy reduces the chance of having interpolants based on elongated distributions of points, which may degrade the accuracy of the interpolation. The details of the subdivision algorithm are given in Algorithm 3.

Algorithm 3: SUBDIVIDE(K) Subdivide local kriging interpolant.

1: $P \leftarrow$ POINTS(K) // Extract set of point-value pairs from the interpolant
2: $\mathbf{M}_2 \leftarrow$ SECONDMOMENT(P) // Compute second moment of point distribution
3: $\lambda_{max} \leftarrow$ MAX(EIGENVALUE$(\mathbf{M}_2)$) // Compute maximum eigenvalue of $\mathbf{M}_2$
4: $\hat{\mathbf{n}} \leftarrow$ EIGENVECTOR(λ_{max}) // Compute eigenvector corresponding to λ_{max}
5: $\mathbf{C} \leftarrow$ CENTROID(K) // Extract the center from the interpolant
6: **for all** $(\mathbf{i}, \mathbf{o}) \in P$ **do**
7: **if** PLANE$(\hat{\mathbf{n}}, \mathbf{C}, \mathbf{i}) \leq 0$ **then**
8: ADDPOINT$(K_1, \mathbf{i}, \mathbf{o})$ // Add point-value pair to the first new interpolant
9: **else**
10: ADDPOINT$(K_2, \mathbf{i}, \mathbf{o})$ // Add point-value pair to the second new interpolant
11: **end if**
12: **end for**
13: **return** (K_1, K_2)

It is important to emphasize that an essential feature of our sampling algorithm is its adaptivity: local kriging interpolants over I are not required to be constructed prior to a simulation. Such construction proceeds in a manner that adapts directly to the evolution of the coarse model. Thus, at a given instant during the course of the simulation, the kriging interpolation extends only over a subset of I. Moreover, as this subset evolves so does the kriging interpolant, following the path through I prescribed by the course model. Maximum benefit of our adaptive sampling approach is realized when the subset remains relatively small with respect to I.

7 Numerical tests

The accuracy of our adaptive sampling algorithm is largely determined by the following four factors:

- The value of the maximum acceptable kriging error estimate (equation (14)), TOL.
- The correlation function, $R(\mathbf{x}, \mathbf{w})$ in the kriging interpolation.
- The number of local kriging interpolants, C_{max}, extracted from the M-tree database.
- The maximum number of point-value pairs in a local kriging interpolant, N_{max}.

In this section, we analyze the influence of these factors on the accuracy of the adaptive sampling process. However, instead of focusing on just the error of the adaptive sampling module alone, we investigate the overall effect of the adaptive sampling on the error induced in the coarse scale model. Recall that we use conventional explicit finite elements in the coarse scale model and the fine scale model is taken as a conventional crystal plasticity material constitutive relation. We compute the error due to the presence of the adaptive sampling at the level of the finite elements. Thus, the error is measured at the level of interest in applications and is amenable to direct physical interpretation.

7.1 Test problem definition

We take pressure driven cylinder expansion as a benchmark problem for assessing the performance of adaptive sampling. A prominent feature of this problem is marked localization of deformation along clearly defined bands, which, ultimately, lead to fracture and fragmentation. Here, we do not attempt to simulate the fracture phenomenon and instead focus solely on the localization. Still, the cylinder expansion problem constitutes an extremely challenging test for the adaptive sampling methodology.

The test sample is a 2.54-cm inner radius steel cylinder with a 0.3-cm thick wall. We model an axial thin section covering one-eighth of the cylinder arc (Figure 4). Plane strain deformation is assumed in the axial direction. Both arc end surfaces, denoted as *A* and *B* in Figure 4, are considered as planes of symmetry. At the onset of the simulation, a uniform pressure of 500 MPa is applied to the interior cylinder surface. The outer cylinder surface remains traction free throughout the entire simulation.

The finite element discretization involves 20 elements through the cylinder wall thickness, 72 elements along the arc and 1 element along the length of the cylinder. The total number of elements employed is $1,440$. We show the discretization employed in the test problem in Figure 5. Parameter studies are facilitated by the modeling of a relatively small section of the cylinder, with substantially larger sections successfully treated using the same techniques in [4].

7.2 Fine model

The constitutive model for the cylinder material captures both thermo-elastic and visco-plastic deformations of a polycrystalline aggregate. We represent the visco-plastic part of the response using a computationally demanding fine-scale model. Thus, the adaptive sampling methodology operates exclusively on the visco-plastic part of the response, with the remainder treated using a conventional continuum mechanics approach. Overall, the material model framework has much in common with models described in [2, 3]. Material models of this sort have evolved out of an extensive literature [24]. By means of a multiplicative decomposition of the deformation gradient

$$\mathbf{F} = \mathbf{V} \cdot \mathbf{R} \cdot \mathbf{F}^{\mathrm{p}}, \tag{25}$$

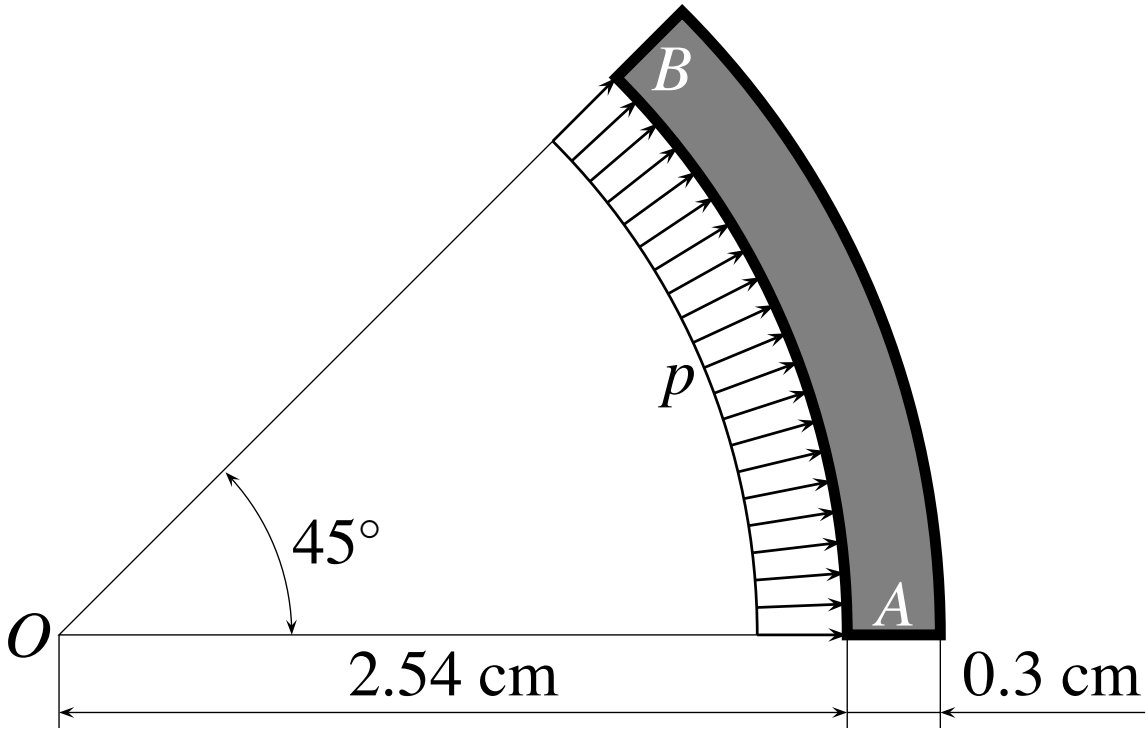


Figure 4: Geometry of the test case problem. O denotes the cylinder axis. The interior cylinder wall is subject to uniform and constant pressure p. A and B denote planes of symmetry.

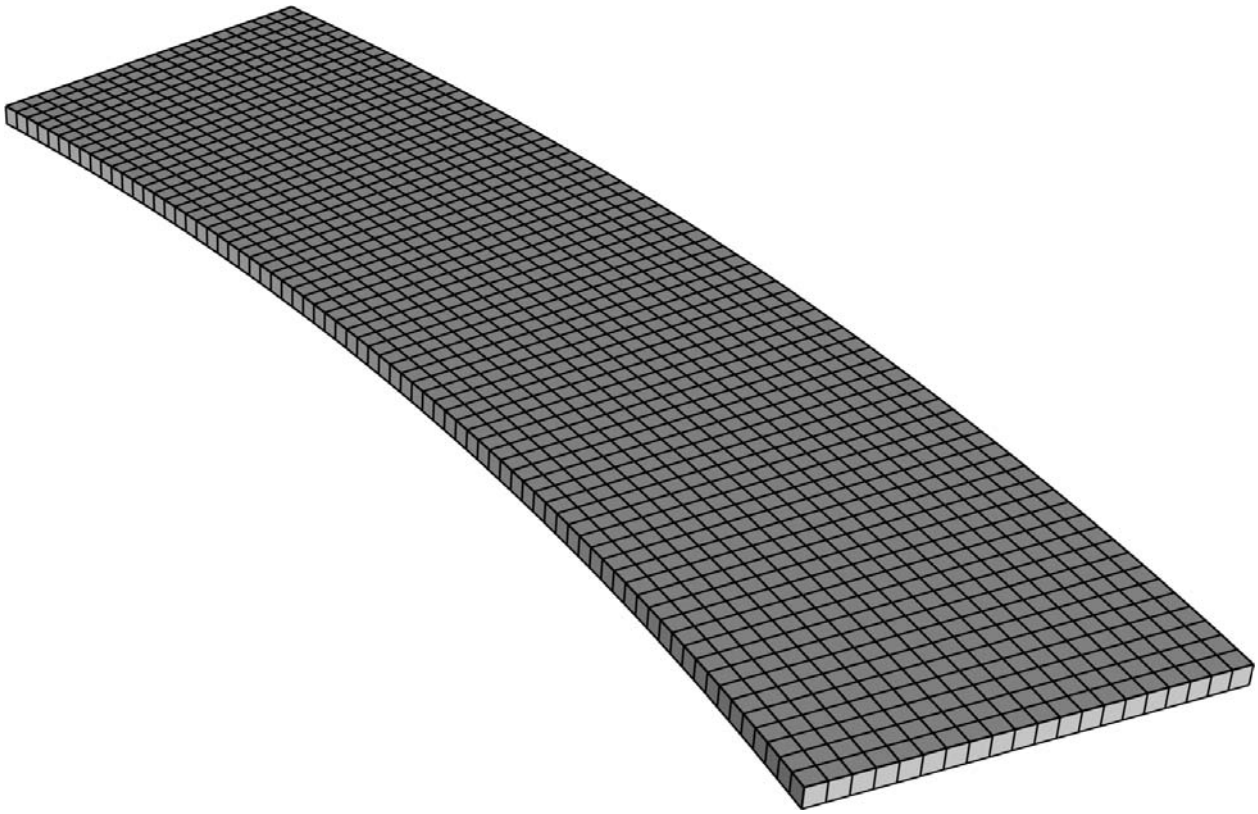


Figure 5: The finite element discretization employed in the test case.

$\mathbf{V}$ captures thermo-elastic stretch, $\mathbf{R}$ captures rotation of the material frame associated with the fine-scale material state, and $\mathbf{F}^{\mathrm{p}}$ captures plastic deformation. The plasticity model is phrased in rate form, with $\bar{\mathbf{L}} = \dot{\mathbf{F}}^{\mathrm{p}} \cdot (\mathbf{F}^{\mathrm{p}})^{-1}$ determined from the fine-scale response. $\bar{\mathbf{L}}$ is customarily referred to as the plastic velocity gradient, although it is not necessarily the gradient of any velocity field. At the fine-scale, we model the plastic response of the polycrystalline material by assuming that all grains in the polycrystal undergo the same deformation rate (symmetric part of $\bar{\mathbf{L}}$) and that each grain deforms plastically by a restricted slip mechanism [24]. Assuming uniform deformation rate gives an upper bound on the stress required to produce a given deformation rate. Other fine-scale models are possible, including self-consistent visco-plastic models [25, 26] or finite element discretization of the polycrystal [39]. Although the computational work associated with the uniform deformation rate model is small compared to that for a finite element simulation of the polycrystal deformation, the fine-scale calculations still dominate the overall computational expense. Further details of the material model may be found in [4].

The individual parameters involved in the point-value relationship for the plasticity model are represented as

$$\{\bar{\mathbf{L}}_{\circ}, g, m, \dot{\gamma}_{\circ}\} = f_{\mathcal{Y}}(\bar{\boldsymbol{\tau}}', h, p, T) \tag{26}$$

in which $\bar{\boldsymbol{\tau}}'$ is the deviatoric part of the Kirchhoff stress, h captures hardening of the material, p is the pressure, T is the temperature, $\bar{\mathbf{L}}_{\circ}$ gives directionality of the plastic velocity gradient, g scales material strength, m is the rate sensitivity, and $\dot{\gamma}_{\circ}$ indicates a reference amount of crystallographic slip activity at the fine-scale. Evaluations of (26) are approximated using kriging models, with an input space dimension of 8 and an output space dimension of 11. Note that $\bar{\boldsymbol{\tau}}'$ has 5 independent components, and $\bar{\mathbf{L}}_{\circ}$ has 8 independent components given that the fine-scale response is isochoric by construction. In this way, equation (26) is a particular instance of a multi-scale response function (1).

With an approximated set of plasticity parameters, the function

$$\bar{\mathbf{L}} = \bar{\mathbf{L}}_{\circ} \left| \frac{\|\bar{\boldsymbol{\tau}}'\|}{g} \right|^{1/m} \tag{27}$$

which is inexpensive to compute, provides the plastic velocity gradient. The same type of scaling relationship is also used by Dawson et al. [15]. Evolution of h depends only on the fine-scale slip activity scaled from $\dot{\gamma}_{\circ}$ so that the plasticity parameters may also be used to update h.

The material parameters are fit to describe the mechanical response of common steels. Crystal lattice orientations in the fine-scale aggregate are sampled from a uniform orientation distribution function. We refer the reader to Barton et al. [4] for the particulars of the material model, as well as the values of the parameters.

7.3 Choice of the stochastic correlation function

The local kriging interpolant (9) relies on the choice of the correlation function R. While numerous options for R have been investigated, the Gaussian exponential form has been shown to provide superior interpolation error properties [28, 29]. Henceforth, we assume that R takes on the well-known functional form of the Gaussian exponential:

$$R(\mathbf{x}, \mathbf{w}) = \exp(-\theta \, ||\mathbf{x} - \mathbf{w}||_2^2), \tag{28}$$

where: $\mathbf{x}, \mathbf{w} \in \mathbb{R}^n$, $|| \bullet ||_2$ represents the L_2-norm in $\mathbb{R}^n$, and $\theta \in \mathbb{R}^+$ is a parameter.

The mathematical underpinnings of the kriging interpolation allow one to tailor individually any parameters involved in R in order to minimize the mean squared error (14). In the case of the Gaussian exponential

correlation function, the minimization procedure encompasses a single parameter, θ. A minimization procedure would most likely require repeated construction of the local kriging interpolant, thus substantially increasing the overall interpolation cost. Moreover, any change of the local kriging interpolant, such as an insertion of a new point into the local kriging interpolant, or its subdivision, would require subsequent recalculation of the optimal θ. In view of the above, we forgo entirely the computation of such optimal θ. In addition, we impose an even more severe restriction requiring that a single value of θ be suitable for all local kriging interpolants composing the kriging interpolation scheme. However, as indicated by the work of Lophaven et al. [28], a relatively wide range of θ may provide sufficient accuracy to warrant the above approach. For sufficiently expensive fine scale evaluations, local optimization of θ may prove effective in reducing total computational work. We may explore this approach in future work.

7.4 Choice of M-tree parameters and algorithmic options

The node capacity for the M-tree database was set to 12 for all nodes in the tree. That is, each node was allowed to contain 12 objects before it was split using the procedure described in Section 5.2.3. This value seemed to give a reasonable tree height and node fan-out for out benchmark problem. However, further study of M-tree performance with respect to this parameter is required and will be a subject of future work.

For the M-tree split policy, we chose relatively simple and inexpensive options. For splitting the root node, we chose a promotion strategy that selected two objects from the existing root node for insertion into the new root node that minimized the overlap of the covering trees for those objects. Specifically, we define the quantity

$$\text{overlap}_{i,j} = \text{dist}(R_i, R_j) - (\text{rad}(R_i) + \text{rad}(R_j)) \tag{29}$$

for each pair of root node entries R_i, R_j ($i \neq j$) and choose two entries R_i and R_j that minimize this value. This choice reduces the number of subtrees that must be traversed from the root node during subsequent object insertions and searches. Since this promotion strategy involves on the order of N^2 distance computations, we chose a simple and cheap procedure for promoting objects at all non-root nodes. In particular, we chose to promote the two objects that are farthest from the parent of the node. Since the array of entries in each node is always sorted with respect to increasing distance from the parent, we promote the last two entries in the array. This strategy requires no distance computations to identify promoted entries and maintains good separation of the subtrees induced by the promotion.

After objects are promoted to the newly-created node, we partition them using the *hyperplane partition* algorithm described in [11]. In this procedure, we first compute the distance between each entry and each of the two promoted objects. Next, we assign each object to the node identified by the closest promoted object. If either node has fewer than some pre-specified minimum fraction of entries, we then re-assign some entries assigned to the "under-full" node. The entries re-assigned are those that are closest to the parent of the "under-full" node. In the end, each node has no less than the minimum fraction of entries. For the results presented here, we chose a minimum fraction of $\frac{1}{2}$. This is referred to as a *balanced* split.

As noted earlier, there exist a variety of alternatives for these M-tree operations and parameters. We chose the options described here based on their simplicity and benefits reported in M-tree studies [11, 12, 13]. In the future, we may explore other options as individual applications warrant.

7.5 Adaptive sampling accuracy metrics

To assess the accuracy of the adaptive sampling process, we compare the effective plastic strain fields obtained with and without the assistance of adaptive sampling. In many applications, especially those involving

catastrophic failure, the effective plastic strain is likely to be one of the most essential features of the solution. Therefore, it is necessary for adaptive sampling to accurately capture both the structure and magnitude of this field.

We plot the effective plastic strain in the test sample obtained without adaptive sampling at the time of 6 μs in Figure 6. As is evident in this figure, the section of the cylinder has undergone substantial inelastic deformation. Moreover, the traces of highly localized deformation, in the form of band-like structures, are clearly visible. Thus, in the subsequent analysis we take the effective plastic strain field, $\varepsilon^f_{ref}(\mathbf{X})$, at 6 μs as the reference. Similarly, $\varepsilon^f(\mathbf{X})$ denotes the effective plastic strain at the time of 6 μs computed with adaptive sampling enabled. $\mathbf{X} \in \mathcal{B} \subset \mathbb{R}^3$ represents a point in the initial configuration, $\mathcal{B}$, of the sample. We define the error field as

$$e(\mathbf{X}) = |\varepsilon^f(\mathbf{X}) - \varepsilon^f_{ref}(\mathbf{X})|. \tag{30}$$

Similarly, we define the relative error field over $\mathcal{B}$

$$e_{rel}(\mathbf{X}) = \frac{|\varepsilon^f(\mathbf{X}) - \varepsilon^f_{ref}(\mathbf{X})|}{\varepsilon^f_{ref}(\mathbf{X})}. \tag{31}$$

In addition to the two error fields, we define their aggregate counterparts obtained as L_∞-norms of appropriate fields over $\mathcal{B}$, i.e.

$$e = \max_{\mathbf{X}\in\mathcal{B}} e(\mathbf{X}) \tag{32}$$

and

$$e_{rel} = \max_{\mathbf{X}\in\mathcal{B}} e_{rel}(\mathbf{X}). \tag{33}$$

Note that the relative error field may always be constructed as our reference effective plastic strain field is uniformly greater than zero at time 6 μs. Moreover, in the realm of finite elements, fields $\varepsilon^f(\mathbf{X})$ and $\varepsilon^f_{ref}(\mathbf{X})$ exist only on a finite set of points in $\mathcal{B}$ (c.f. [21]), in which case the evaluation of equations (32) and (33) is carried exclusively by means of this point set.

7.6 Adaptive sampling performance metrics

We estimate the performance of the adaptive sampling methodology by means of an evaluation fraction, m_{ef}, defined as

$$m_{ef} = \frac{N_{eval}}{N_{total}}, \tag{34}$$

where N_{eval} represents the total number of the fine model evaluations in the benchmark adaptive sampling simulation and N_{total} is the total number of requests for the value of the response function. According to this definition, values of m_{ef} fall between 0 and 1. A value of m_{ef} near 1 implies little or no performance benefit from adaptive sampling. As m_{ef} decreases toward, 0 the efficiency of the adaptive sampling process and the performance gain increases.

In addition to the evaluation fraction, we investigate the performance of the adaptive sampling methodology from the viewpoint of wall-clock speedup. This is an important issue, as the low evaluation fraction does not guarantee that there will be an overall execution time benefit. We apply the following fairly conventional definition of speedup

$$m_{et} = \frac{t_{AS}}{t}. \tag{35}$$

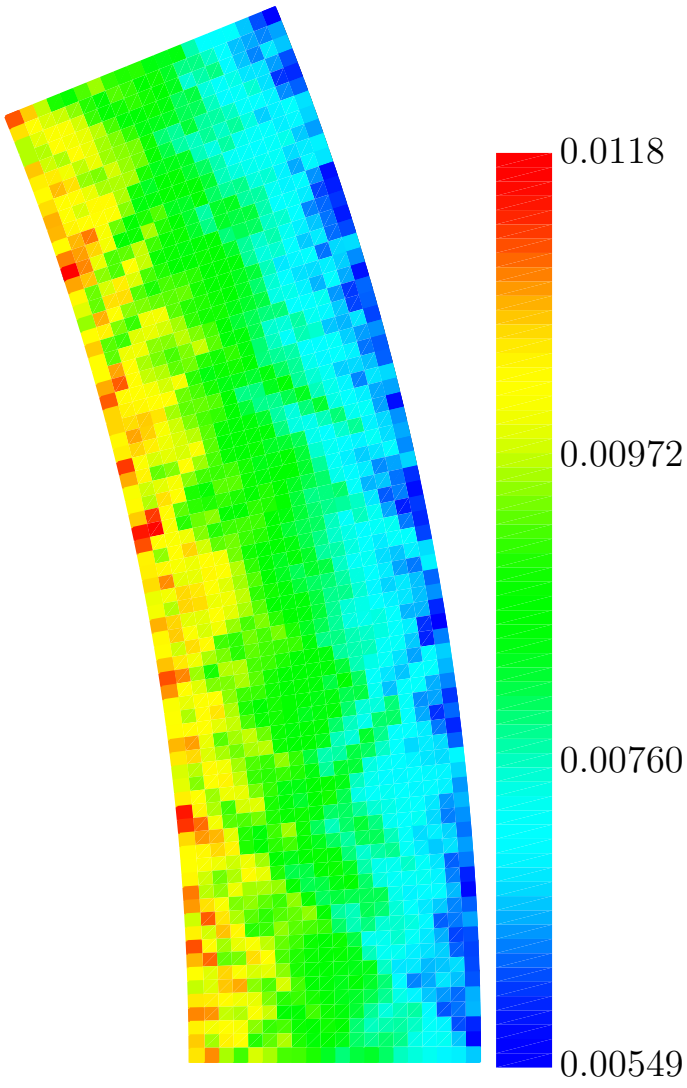


Figure 6: Effective plastic strain at 6 μs.

Here, t_{AS} is the execution time of the benchmark application with adaptive sampling enabled, and t-the execution time of the benchmark application without adaptive sampling. It is important to emphasize that the speedup is solely due to the application of the adaptive sampling methodology and it does not include any benefits from parallel execution. All the speedup runs were performed on a workstation with a single Intel-Xeon 2.3GHz processor.

7.7 Numerical results

We now describe the results of our studies of the adaptive sampling accuracy and computational efficiency. We begin with an investigation of the influence of the value of parameter θ in the Gaussian correlation function and the requested tolerance TOL on the error and the evaluation fraction. Subsequently, for a chosen (θ, TOL)-pair we analyze the effect of the number of local kriging interpolants extracted from the metric-tree database, C_{max}. Finally, once the optimal C_{max} is identified, we study the influence of the local kriging interpolant size, N_{max}.

7.7.1 The effect of θ and TOL

θ and TOL constitute two essential parameters within the adaptive sampling methodology. We investigate the effect of both of these parameters on the error, e, and evaluation fraction, m_{ef}. In our study, we explore the values of θ in the range $10^1 - 10^9$, whereas TOL varies between 10^{-2} and 10^{-5}. C_{max} is set to 6, and N_{max} to 4.

We plot $e(\text{TOL}, \theta)$ in Figure 7 and $m_{ef}(\text{TOL}, \theta)$ in Figure 8. The white triangular area in the lower left corner of both plots represents the domain range inaccessible to our adaptive sampling implementation due to excessive computer memory requirements.

The large red region in the error plot represents e at approximately 10^{-3}. This region is surrounded by the areas of steep error decay universally throughout the entire range of θ. The dependence of e on θ is more complex. θ in the range $10^7 - 10^9$ provides very little in terms of error reduction from the plateau value. This behavior is clearly different in the $10^1 - 10^7$ range, where a rapid decrease of the error can be observed. Overall, the recorded error is within the $10^{-7} - 10^{-3}$ range.

Clearly, the adaptive sampling methodology possesses the ability to properly control the overall error. Moreover, as desired, the error is a monotonically increasing function of tolerance parameter TOL universally over the entire range of θ values. It is also worth noting that our choice of error metric using the equivalent plastic strain, provides essentially an integral view of the entire deformation history. Hence, the capacity of adaptive sampling to attain the error levels below 10^{-3} is truly noteworthy.

The dependence of error on θ, a parameter controlling the support of the correlation function, R, appears non-monotonic. Two ranges can be clearly identified. In the first range, $\theta \in [10^1, 10^7]$, for assumed TOL value, the increase in θ leads to a marked rise in the magnitude of error. However, this trend is reversed in the second range, $\theta \in [10^7, 10^9]$, where the error is marginally reduced with increasing θ. The behavior of error in the first range is a consequence of the increasingly smaller support of the Gaussian exponential (28) as θ increases, gradually lessening the correlation between any two points in a local kriging interpolant. This weaker correlation reduces the benefit of kriging interpolation towards plain linear regression, and ultimately yields higher interpolation errors. We believe that $\theta = 10^7$ constitutes a threshold value above which the contribution of the term in the kriging interpolant is rapidly diminishing. Therefore, it is likely that the error may be somewhat reduced but at the expense of considerably higher evaluation fraction, as evident from Figure 8. For the particular test problem employed here, it appears desirable to refrain from using θ larger than the above threshold value.

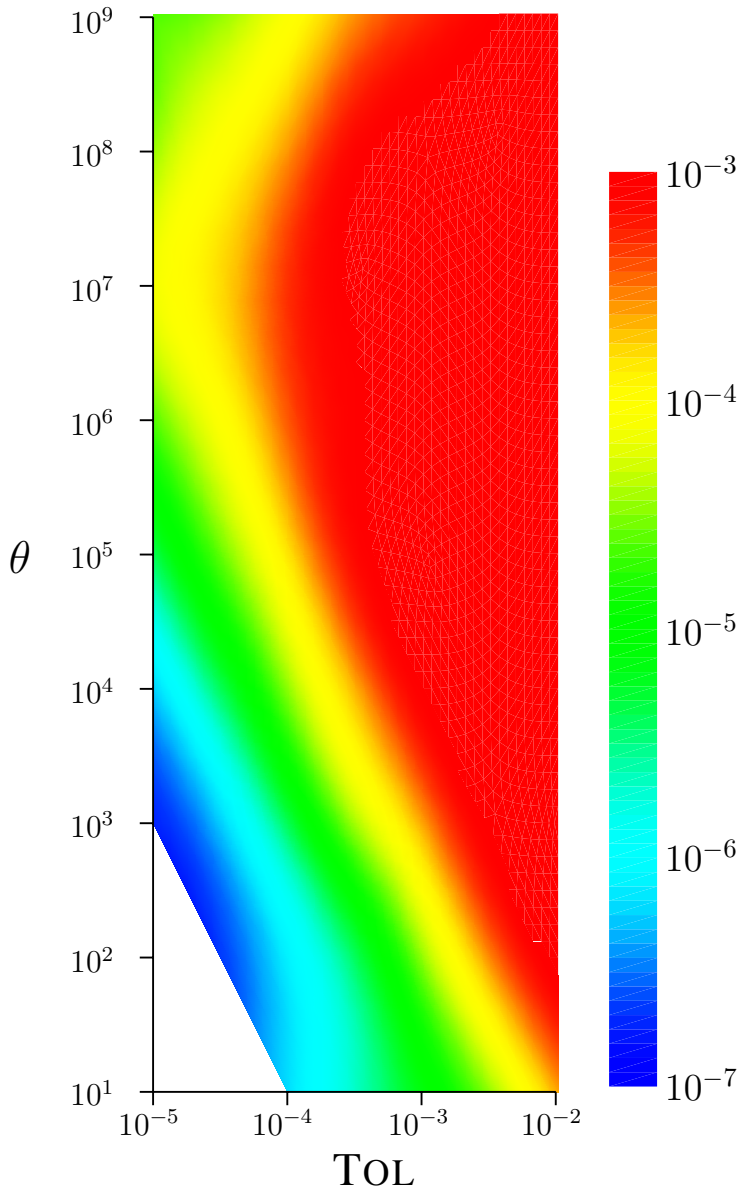


Figure 7: Error e as a function of the requested tolerance TOL and the parameter θ of the Gaussian exponential correlation function.

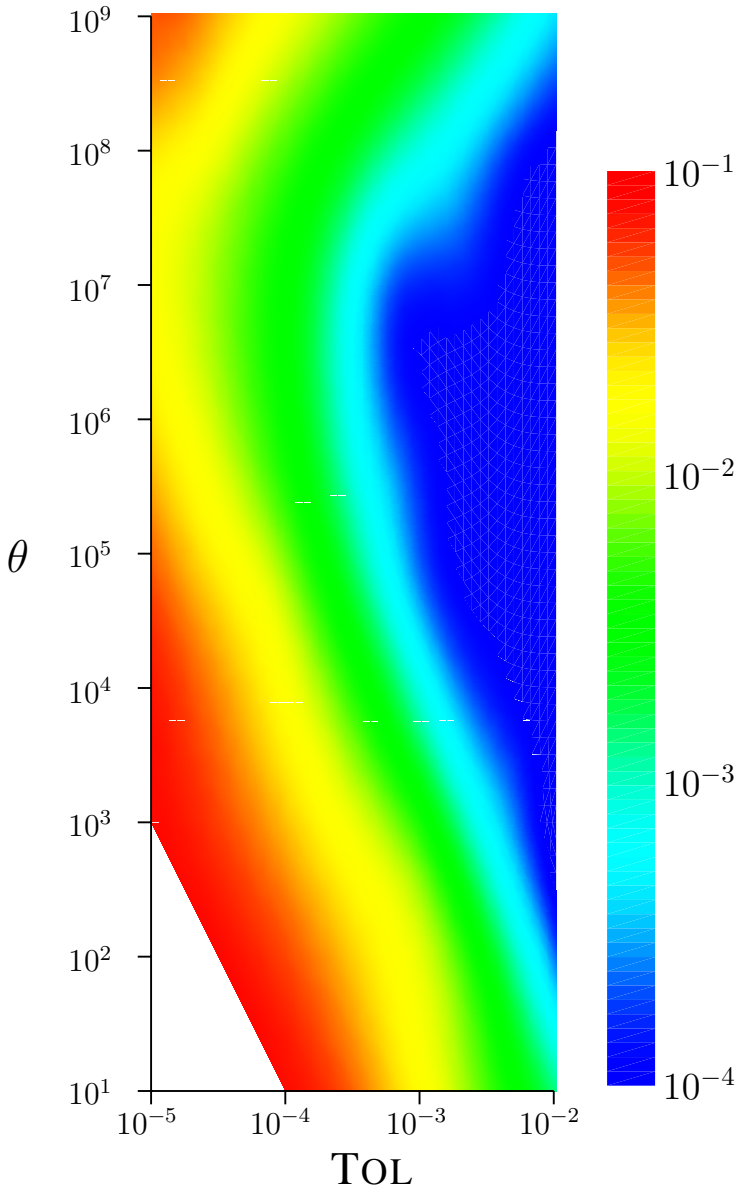


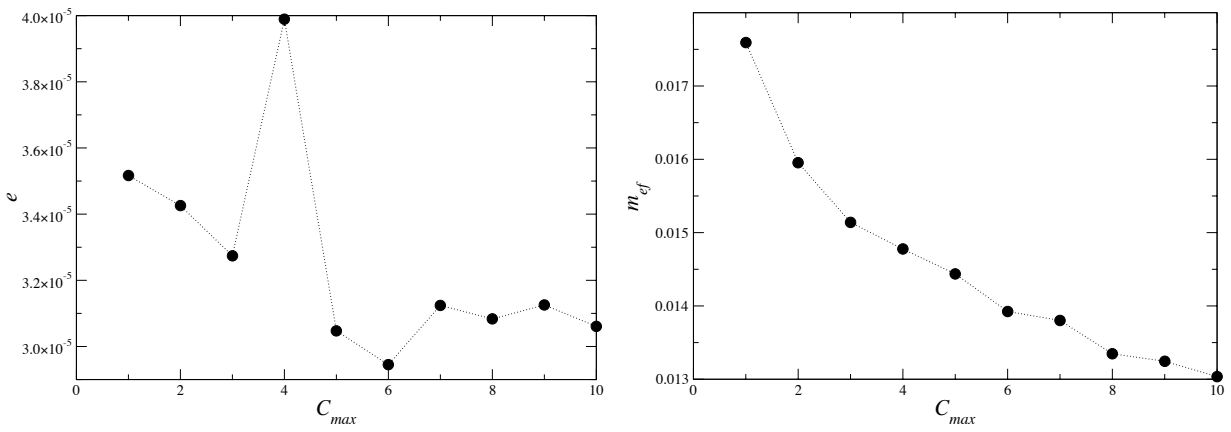
Figure 8: Evaluation fraction m_{ef} as a function of the requested tolerance TOL and the parameter θ of the Gaussian exponential correlation function.

Predictably, the m_{ef} plot displays features similar to the error plot, with m_{ef} values falling in the $10^{-4} - 10^{-1}$ range. However, regions of lower evaluation fraction are obviously correlated with regions of higher error.

The contours of e and m_{ef} in Figures 7 and 8 clearly indicate that neither TOL nor θ alone possesses the capacity of controlling the performance of adaptive sampling. For example, for the following two points in the space of pairs (TOL, θ), $P_1 = (10^{-3}, 10^1)$ and $P_2 = (10^{-4}, 10^4)$, we have $e(P_1) \approx e(P_2)$, as well as, $m_{ef}(P_1) \approx m_{ef}(P_2)$. Hence, in practical applications, a number of numerical tests may be required in order to provide some appropriate values of both parameters. However, as is evident from our study, such values are readily available as a broad range of acceptable values exists.

7.7.2 The effect of C_{max}

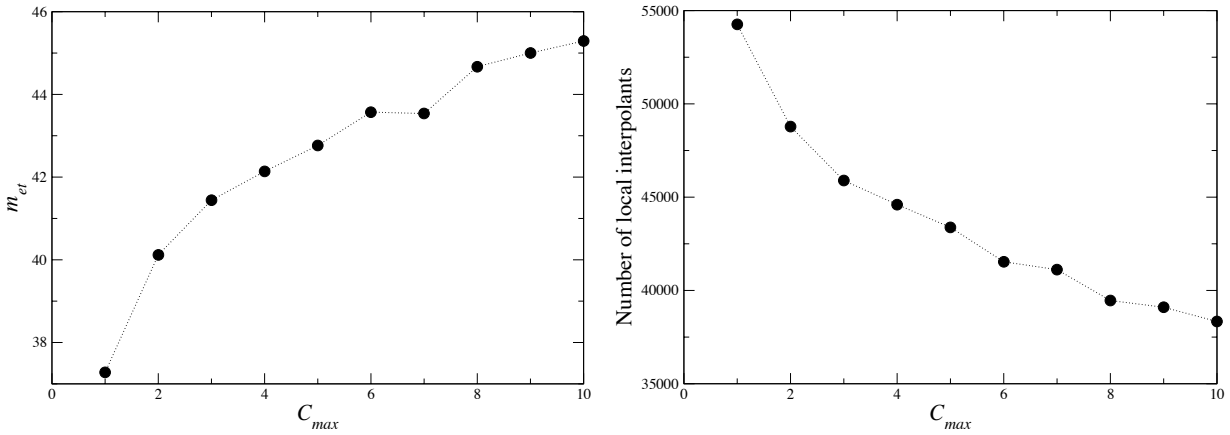
On the basis of the previous investigation, we adopt $\theta = 10$ and $\text{TOL} = 10^{-3}$ as a reference for the parametric study of the influence of the number of local kriging interpolants extracted from the M-tree database during a kNN-search, C_{max}. The goal is to choose C_{max} large enough to maintain a desired level of accuracy and reap substantial computational savings from relatively low evaluation fraction, but not so large that the cost of M-tree searching becomes excessive. As in the preceding study, the size of the local kriging interpolant, N_{max}, remains equal to 4.



(a) (b)

Figure 9: Error (a) and evaluation fraction (b) as a function of the number of local kriging interpolants retrieved from the metric tree database.

Figure 9 contains plots of e and m_{ef} both as a function of C_{max}. There appears to be no clear trend present in the character of e, although the values of e seem to stabilize somewhat for larger C_{max}. Overall, however, the changes in e remain relatively small, indicating an accurate error control over the wide range of C_{max}. Such behavior is not unexpected, as the adaptive sampling algorithm selects the closest local kriging interpolant satisfying the tolerance requirement. It is quite likely that e could be further reduced by choosing the local kriging interpolant providing the smallest interpolation error estimate among all C_{max} local interpolants. However, such a modification would entail a computation of the error estimate for all local interpolants involved and, ultimately, could substantially increase the computational cost of the entire adaptive sampling algorithm.



(a) (b)

Figure 10: The wall-clock speedup (a) and the number of local kriging models (b) as a function of the number of local kriging interpolants retrieved from the metric tree database.

The monotonically decreasing trend of the m_{ef} as a function of C_{max}) makes apparent the benefit of considering more than one local kriging interpolant. Obviously, having a larger pool of potential candidates increases chances for locating a suitable local interpolant and avoiding the expense of explicit response function evaluation, which in turns lowers the evaluation fraction. Hence, the measured reduction of the evaluation fraction due to C_{max} amounts to 26%. This substantial reduction of the evaluation fraction allows us to realize remarkable execution-time benefits, as evident in Figure 10(a), where we plot the wall clock speedup, m_{et}, as a function of C_{max}. m_{et} is a monotonically increasing function, with most gains realized for small C_{max} values as the trend toward saturation may clearly be observed for the larger values. Over the entire range of C_{max} explored the speedup varies between 37 and 45, a notable result. Finally, we present a plot of the total number of local kriging interpolants as a function of C_{max} in Figure 10(b). Not unexpectedly, the number of local kriging interpolants follows ostensibly the trends displayed by the evaluation fraction.

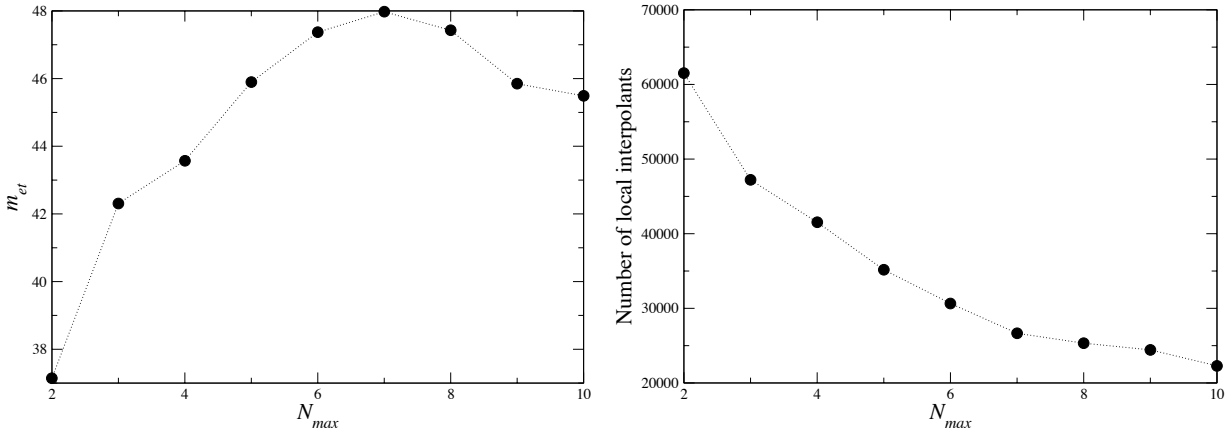
7.7.3 The effect of N_{max}

Next, we investigate the influence of the maximum allowed size of the local kriging interpolant, N_{max}, on the performance of adaptive sampling. We follow the same methodology as in the previous studies and take $\theta = 10$ and $\text{TOL} = 10^{-3}$. Moreover, we fix C_{max} at 6 as the value yielding the most accurate test problem solution among all investigated. We study the values of N_{max} between 2 and 10, as $N_{max} = 1$ is simply equivalent to the conventional linear regression. In addition, based on our computer experiments, the performance of adaptive sampling based solely on the linear regression is truly inferior with respect to kriging, especially in terms of the required number of local interpolants.

The error and evaluation fraction as a function of N_{max} are plotted in Figure 14. The error appears only weakly dependent on N_{max}, and remains essentially uniform over the entire range under investigation. Thus, for the problems we consider here, the use of local interpolants with larger support does not lead to substantial gains in the overall accuracy. Such behavior serves as yet another indication of the appropriate error control of adaptive sampling, as the size of the set of points supporting a local kriging interpolant

does not affect the accuracy of the interpolation. The accuracy is controlled most directly by the requested interpolation tolerance.

The behavior of the evaluation fraction is largely in line with our expectations, as the increase in N_{max} is accompanied by a monotonic decrease in the evaluation fraction. Obviously, the increase in the support of local kriging interpolants improves the ability of such models to provide more accurate interpolated values even outside of the support point-set. In addition, larger local interpolation models may provide more uniform coverage of I. We finally remark that for large values of parameter θ, increasing N_{max} may have a detrimental effect on the interpolation accuracy and such a combination should most likely be avoided.



(a) (b)

Figure 11: The wall-clock speedup (a) and the number of local kriging models (b) as a function of the maximum size of the local kriging interpolant.

In the context of the error incurred due to adaptive sampling, it is also instructive to investigate the distribution of error over the simulation domain, instead of aggregate error measure e. We plot the error and relative error fields due to adaptive sampling in Figures 12 and 13, respectively. The value of $N_{max} = 4$ is used to generate the fields. Regions of the computational domain incurring higher errors are roughly correlated with those undergoing substantial plastic deformation. Moreover, the spatial structure of the relative error field closely mirrors that of the error field. The maximum value of the relative error of just 3.14×10^{-3} serves as a good indication of truly remarkable error control. Here, we emphasize again that the effective plastic strain may be regarded as an internal history variable being integrated from its initial value of zero over the entire course of the simulation. Thus, the high fidelity of the solution in terms of the effective plastic strain suggests high accuracy with respect to other variables.

The overall gains due to adaptive sampling are a function of the relative expense of carrying out the interpolation versus executing the fine model. However, the ultimate adaptive sampling benefits are best expressed in the execution speedup. We plot the effects of parameter N_{max} on the execution speedup in Figure 11(a). Clearly, adaptive sampling methodology offers tremendous computational savings universally over the range of N_{max} investigated. The attained speedup ranges between 37 and 48, with $N_{max} = 7$ yielding the maximum speedup. Two factors influence the emergence of this maximum. First, the increasing size of local kriging interpolants increases the cost of their construction and evaluation. Second, the

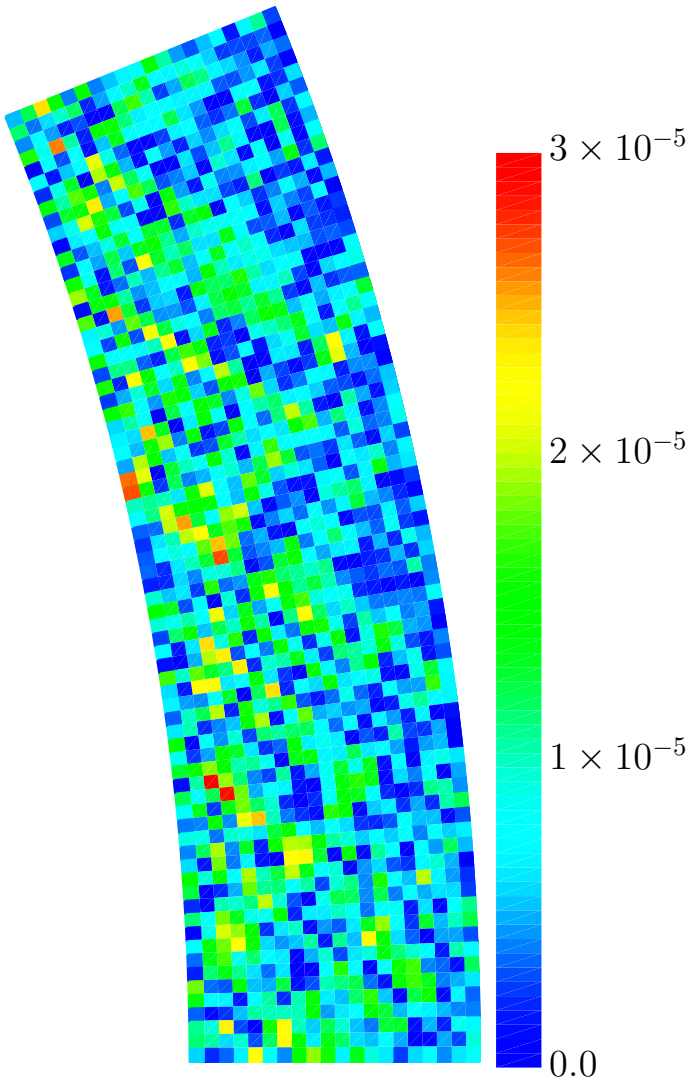


Figure 12: The distribution of error, e.

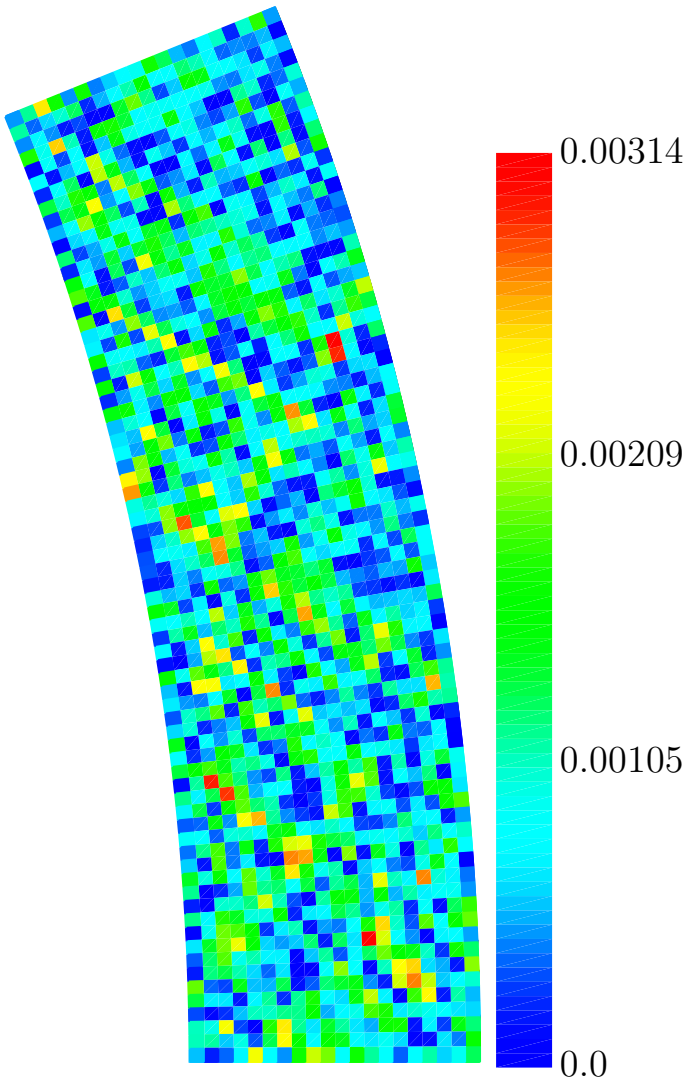
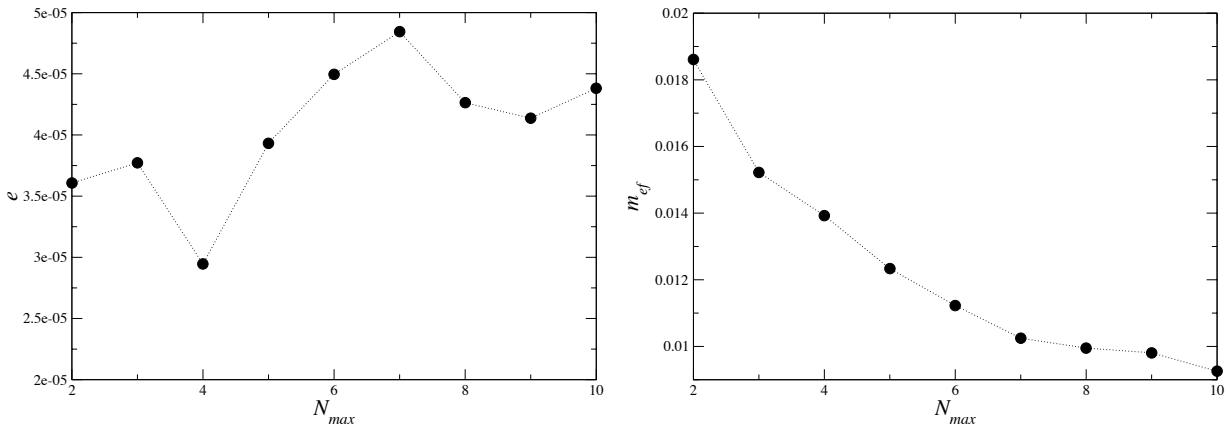


Figure 13: The distribution of relative error, e_{rel}.



(a) (b)

Figure 14: Error (a) and evaluation fraction (b) as a function of the maximum size of the local kriging interpolant.

decreasing evaluation fraction lowers the expense of fine model evaluation as well as the M-tree database searching. The maximum is a result of a competition between these two factors. To further emphasize our point, we plot the total number of local kriging interpolants stored in the M-tree database in Figure 11(b). As is evident from the plot, the size of the M-tree database decreases measurably with N_{max} providing further support to our analysis.

8 Summary

We presented an adaptive sampling methodology for hierarchical multi-scale simulation. In our approach, we rely on the application of the local kriging interpolation to substantially reduce the number of required evaluations of the finer-scale response function. The set of points that define the kriging interpolation is not required to form any particular spatial arrangement and is allowed to evolve continuously under the direction of the coarse-scale model. This point-set is divided into a collection of subsets, which are used to construct local kriging interpolants. The resulting interpolation scheme is therefore fully unstructured and adaptive. Moreover, the adaptivity is completely local, thus no extensive modifications to the existing kriging interpolation scheme are necessary when new points are inserted. However, this locality imposes stringent performance requirements on the management of the underlying collection of local kriging interpolants. We utilize an M-tree database which supports rapid insertion and retrieval of suitable local kriging interpolants.

The performance of our adaptive sampling methodology has been investigated in the context of a two-level multi-scale hierarchy involving a conventional finite element model at the coarse scale and a constitutive material response at the fine scale. Our studies indicate that adaptive sampling is capable of accurate error control over a wide range of parameters. More importantly, decreases in the interpolation tolerance are uniformly accompanied by higher accuracy of the solution at the expense of a higher fine-scale model evaluation fraction. Neither the number of local kriging interpolants extracted from the M-tree database nor the maximum allowed number of points associated with local kriging interpolants appear to strongly influence the accuracy of the solution. However, judicious choices of these parameters lead to a substantial reduction

of the evaluation fraction. As desired, this reduction yields significant benefits in terms of the simulation execution time. For the test problem considered here, we have demonstrated a wall clock execution time speedup factor of 48.

The adaptive sampling methodology presented here may be extended to multi-scale hierarchies comprising more than two levels. A two-level hierarchy, such as the one studied here, constitutes a fundamental building block of a more general multi-scale simulation. In such a case, the overall performance benefit from adaptive sampling would be a composition of gains at each level in the hierarchy, leading potentially to truly exceptional computational savings.

Acknowledgments

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory (LLNL) under Contract No. W-7405-Eng-48. The project 04-ERD-102 was funded by the Laboratory Directed Research and Development Program at LLNL.

References

[1] P. Alfeld. Scattered data interpolation in three or more variables. In *Mathematical methods in computer aided geometric design*, pages 1–33. Academic Press Professional, Inc., San Diego, CA, USA, 1989. ISBN 0-12-460515-X.

[2] A. Arsenlis, N. R. Barton, R. Becker, and R. Rudd. Generalized *in situ* adaptive tabulation for constitutive model evaluation in plasticity. *Comput. Methods Appl. Mech. Engrg*, 196(1–3):1–13, 2006.

[3] N. R. Barton, D. J. Benson, and R. Becker. Crystal level simulations using Eulerian finite element methods. In *Materials processing and design: modeling, simulation and applications*, volume 712 of *AIP Conference Proceedings*, pages 1624–1629, Columbus, Ohio, 2004. NUMIFORM.

[4] N. R. Barton, J. Knap, A. Arsenlis, R. Becker, R. Hornung, and D. Jefferson. Embedded polycrystal plasticity and adaptive sampling. URL http://dx.doi.org/10.1016/j.ijplas.2007.03.004. to appear, 2007.

[5] A. J. Beaudoin, P. R. Dawson, K. K. Mathur, U. F. Kocks, and D. A. Korzekwa. Application of polycrystalline plasticity to sheet forming. *Comput. Methods Appl. Mech. Engrg*, 117:49–70, 1994.

[6] T. Belytschko, W.K. Liu, and B. Moran. *Nonlinear finite elements for continua and structures*. Wiley, New York, NY, USA, 2000.

[7] R. W. Bilger, S. B. Pope, K. N. C. Bray, and J. F. Driscoll. Paradigms in turbulent combustion research. *Proc. Combust. Inst*, 30:21–42, 2005.

[8] C. Bohm, S. Berchtold, and D. Keim. Searching in high-dimensional spaces – index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322–373, 2001.

[9] B. D. Bojanov, H. A. Hakopian, and A. A Saakian. *Spline Functions and Multivariate Interpolations*. Kluwer Academic Publishers, Dordrecht, May 1993.

[10] H. S. Chung and J. J. Alonso. Using gradients to construct cokriging approximation models for high-dimensional design optimization problems. In *AIAA 40th Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 14-17, 2002.

[11] P. Ciaccia, M. Patella, and P. Zazula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd Conference on Very Large Database (VLDB '97)*, Athens, Greece, 1997.

[12] P. Ciaccia, M. Patella, and P. Zazula. A cost model for similarity queries in metric spaces. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '98)*, Seattle, WA, 1998.

[13] P. Ciaccia, M. Patella, and P. Zazula. Processing complex similarity queries with distance-based access methods. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT '98)*, Valencia, Spain, 1998.

[14] N. A. C. Cressie. *Statistics for Spatial Data*. Wiley, New York, revised edition, 1993.

[15] Paul R. Dawson, Donald E. Boyce, Ryan Hale, and John P. Durkot. An isoparametric piecewise representation of the anisotropic strength of polycrystalline solids. *Int. J. Plasticity*, 21:251–283, 2005.

[16] V. Gaede and O. Gunther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[17] M. Gasca and T. Sauer. On the history of multivariate polynomial interpolation. *J. Comput. Appl. Math.*, 122(1–2):23–35, 2000.

[18] M. Gasca and T. Sauer. Polynomial interpolation in several variables. *Adv. Comput. Math.*, 12(4): 377–410, 2000.

[19] N. M. Ghoniem, E. P. Busso, N. Kioussis, and H. Huang. Multiscale modelling of nanomechanics and micromechanics: an overview. *Philosophical Magazine*, 83(31):3475–3528, 2003.

[20] L. Gu. Moving kriging interpolation and element-free galerkin methods. *Internat. J. Numer. Meths. Engrg.*, 56(1):1–11, 2003.

[21] T.J.R. Hughes. *The Finite Element Method: linear static and dynamic finite element analysis*. Dover Publications, Inc., Mineola, NY, USA, 2000.

[22] E. Isaaks and R. M. Srivastava. *An Introduction to Applied Geostatistics*. Oxford University Press, New York, 1989.

[23] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, and C. Theodoropoulos. Equation-free, coarse-grained multiscale computation: enabling microscopic simulators to perform system-level analysis. *Comm. Math. Sci*, 1(4):715–762, 2003.

[24] U. F. Kocks, C. N. Tomé, and H.-R. Wenk, editors. *Texture and anisotropy: preferred orientations in polycrystals and their effect on materials properties*. Cambridge University Press, 1998.

[25] R. Lebensohn. Modelling the role of local correlations in polycrystal plasticity using viscoplastic self-consistent schemes. *Modelling Simul. Mater. Sci. Eng.*, 7:739–746, 1999.

[26] R. A. Lebensohn and C. N. Tomé. A self-consistent viscoplastic model: prediction of rolling textures of anisotropic polycrystals. *Mater. Sci. Eng.*, A175:71–82, 1994.

[27] W. K. Liu, E. G. Karpov, S. Zhang, and H. S. Park. An introduction to computational nanomechanics and materials. *Comput. Methods Appl. Mech. Engrg*, 193:1529–1578, 2004.

[28] S.N. Lophaven, H.B. Nielsen, and J. Søndergaard. DACE-A Matlab Kriging Toolbox, Version 2.0. Technical Report MM-REP-2002-12, IMM, Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, 2002.

[29] S.N. Lophaven, H.B. Nielsen, and J. Søndergaard. Aspects of the matlab toolbox DACE. Technical Report IMM-REP-2002-13, IMM, Informatics and Mathematical Modelling, The Technical University of Denmark, Lyngby, 2004.

[30] J. Lubliner. *Plasticity theory*. Macmillan, New York, NY, USA, 1990.

[31] M. D. Morris, T. J. Mitchell, and D. Ylvisaker. Bayesian design and analysis of computer experiments: Use of derivatives in surface prediction. *Technometrics*, 35(3):243–255, 1993.

[32] M. Ortiz, A. M. Cuitino, J. Knap, and M. Koslowski. Mixed atomistic-continuum models of material behavior: the art of transcending atomistics and informing continua. *MRS Bulletin*, 26(3):216–221, 2001.

[33] M. Ortiz and R. Phillips. Nanomechanics of defects in solids. *Advances in applied mechanics*, 36: 1–79, 1999.

[34] P. Perzyna. Internal state variables description of dynamic failure in solids. *Int. J. Solids Struct*, 22: 797–818, 1986.

[35] R. Phillips. Multiscale modeling in the mechanics of materials. *Current Opinion in Solid State & Materials Science*, 3(6):526–532, 1998.

[36] S. B. Pope. Computationally efficient implementation of combustion chemistry using in situ adaptive tabulation. *Combustion Theory and Modelling*, 1(1):41–63, 1997.

[37] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 71–79, San Jose, CA, 1995.

[38] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

[39] G. B. Sarma and P. R. Dawson. Effects of interactions among crystals on the inhomogeneous deformations of polycrystals. *Acta Mater.*, 44(5):1937–1953, 1996.

[40] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. *Proceedings of the 1968 23rd ACM national conference*, pages 517–524, 1968.

[41] M. A. Singer and S. B. Pope. Exploiting ISAT to solve the reaction-diffusion equation. *Combust. Theory Modelling*, 8(2):361–383, 2004.

[42] S. Taing, AR Masri, and SB Pope. pdf calculations of turbulent nonpremixed flames of H 2/CO 2 using reduced chemical mechanisms. *Combustion and Flame*, 95(1):133–150, 1993.

[43] C. Theodoropoulos, Y. H. Qian, and I. G. Kevrekidis. "Coarse" stability and bifurcation analysis using time-steppers: A reaction-diffusion example. *Proc. Nat. Acad. Sci*, 97(18):9840–9843, 2000.

[44] S. K. Thompson and G. A. F. Seber. *Adaptive sampling*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[45] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Inf. Proc. Lett.*, 40 (4):175–179, 1991.

[46] J. M. Ver Hoef and N. A. C. Cressie. Multivariable spatial prediction. *Math. Geology*, 25(2):219–240, 1993.

[47] H. Wackernagel. *Multivariate Geostatistics; An Introduction with Applications*. Springer, Berlin, 2nd edition edition, 1998.

[48] R. J. Yang, N. Wang, C. H. Tho, and J. P. Bobineau. Metamodeling development for vehicle frontal impact simulation. *ASME J. Mech. Des.*, 127(5):1014–1020, 2005.

[49] P. Zezula, P. Ciaccia, and F. Rabitti. M-tree: A dynamic index for similarity queries in multimedia databases. Technical Report TR 7, Laboratory of Distributed Multimedia Information Systems and Applications, Technical University of Crete, Greece, 1996.